# Lifting-based Subdivision Wavelets with Geometric Constraints

QIN, Guiming

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Philosophy
in
Computer Science and Engineering

The Chinese University of Hong Kong
August 2010

Thesis Committee

Professor Jia, Jiaya (Chair)

Professor SUN, Hanqiu (Thesis Supervisor)

Professor WONG Tien Tsin (Committee Member)

Professor BACIU George (External Examiner)

# Abstract

The techniques of high-detailed 3D models developed in recent years demand efficient multiresolution representations of polygonal meshes. They are widely used in many fields such as computer animation, data compression, digital geometric processing and so on. This thesis concentrates on explicit formulations for B-spline curves and biorthogonal subdivision wavelet transform approaches and techniques based on lifting schemes with geometric constraints. The main contributions of this thesis include the following several aspects: explicit computation and matrix formulations; dual subdivision wavelet transform; biorthogonal wavelet transforms based on lifting scheme with geometric constraints.

An explicit computing formula for general recursion schemes is proved by mathematical induction first, and then the explicit formulae of the de Boor algorithms for computing non-uniform B-spline curves and their derivatives are proved; finally an explicit symbol matrix representation for non-uniform B-spline curves of arbitrary degree and its transformation to the polynomial space spanned by the common power basis are presented based on our explicit computing formula, implemented in Matlab programming language to make more efficiently computing non-uniform B-spline curves and surfaces of arbitrary degree and their derivatives by using Horner担 scheme available.

This thesis overcomes the difficulty of constructing the biorthogonal wavelet transforms based on the dual subdivision. In our biorthogonal wavelet construction based on the dual subdivision, the centroids of the V-faces corresponding to the old vertices are selected as the virtual vertices associated with the scaling functions to construct the scaling space. The lifting scheme is used to guarantee the fitting quality of the wavelet transform, and a local orthogonalization is introduced with a discrete inner product operation to improve the computation efficiency. The presented dual-subdivision-based wavelet construction is proven to be stable and effective by the experimental results.

we present novel geometrically constrained Loop, Catmull-Clark and Doo-Sabin subdivision wavelet transforms. The geometrically constrained subdivision-based wavelet transforms can preserve several geometric constraints, such as position, high order derivative and isoparametric curve constraints, and improve the geometric modeling ability of 3D multi-resolution models with polygonal meshes. Because the lifting operators we designed with geometric constraints only work in the local areas of the control mesh, it is very convenient and efficient for a broad range of interactive applications of 3D multiresolution models of massive data sets. The presented subdivision wavelet construction approaches based on the lifting schemes with geometric constraints are also proven to be stable and effective by the experimental results.

# 摘要

近年来，3D 多分辨率几何造型技术得到飞速发展，它在计算机动画、数据压缩和数字几何处理等许多领域都有十分广阔的应用前景。本文主要研究 B 样条曲线曲面的显式计算和基于几何约束 lift 技术的双正交细分小波变换及其多分辨率几何造型技术；其主要贡献包括以下几个方面：B 样条曲线曲面的显式计算及其矩阵公式；Doo-Sabin 细分小波构造；基于几何约束 lift 技术的双正交细分小波变换及其多分辨率造型算法。

首先，提出了一个通用递归方法的显示计算公式,并用数学归纳法给出了它的证明过程；然后，给出了非均匀 B 样条曲线及其导数的 de Boor 算法的显式计算公式，以及任意阶非均匀 B 样条曲线的显式矩阵符号表示，并给出了证明；也给出了 B 样条基函数到多项式幂基函数的转换公式;还给出了任意价非均匀 B 样条曲线曲面的矩阵符号计算的 Matlab 程序，使得用 Horner 格式快速地计算任意次非均匀 B 样条曲线曲面及其导数成为可能。

对偶细分，例如 Doo-Sabin 细分，在每次细分后，原有的控制点会消失，每个控制点被一个 V 面取而代之。这对于基于对偶细分的双正交小波变换的构造是一个极大的挑战。本文针对典型的对偶细分格式——Doo-Sabin 细分的特点，克服了基于对偶细分的双正交小波变换的困难。在我们的基于对偶细分的双正交小波构造算法中，通过选择 V 面的中心作为与相关尺度函数相对应的虚拟控制点来构造双正交小波，用 lifting 操作来改善曲面的拟合质量，引入离散内积及局部正交化技术来提高计算效率。实验证明，我们的基于对偶细分的双正交小波变换算法是一种稳定、高效的算法。

为了更好地表现复杂 3D 模型的多分辨率几何造型特点并保持指定的几何特征，本文提出了基于几何约束 Lifting 技术的 Loop 细分小波、Catmull-Clark 细分小波和 Doo-Sabin 细分小波变换方法。这些满足几何约束的细分小波变换能够在多分辨率造型过程中保持指定的位置、一阶或高阶导数、以及等参数曲线约束等几何特征，提高了细分小波的多分辨率造型能力。实验证明，我们的基于几何约束 Lifting 技术的双正交细分小波变换算法也是稳定、高效的算法。

# Contents

# 1 Introduction

## 1.1 B splines and B-splines surfaces

B-splines are defined in terms of a knot sequence $t:=(t_j)$, meaning that

$$\cdots \leqslant t_j \leqslant t_{j+1} \leqslant \cdots$$

The $j^{th}$ B-spline of order 1 for the knot sequence is the characteristic function of the half-open interval $[t_j \cdots t_{j+1})$, i.e., the function given by the rule:

$$B_{j1}(x) := B_{j,1,t}(x) := \begin{cases} 1, & if\ t_j \le x \le t_{j+1} \\ 0, & otherwise \end{cases}$$

Note that each of these functions is piecewise constant, and that:

$$\sum_j B_j(x) = 1, \quad \inf_j t_j < \sup_j t_j.$$

From these first order B-splines, B-splines of higher order can be derived inductively by the B-splines recurrence. The $j^{th}$ B-spline of order $k>1$ for the knot sequence t is:

$$B_{jk} := B_{j,k,t} := \omega_{jk} B_{j,k-1} + (1-\omega_{j+1,k}) B_{j+1,k-1},$$

With $\omega_{jk}(x) := \omega_{j,k,t}(x) := \dfrac{x - t_j}{t_{j+k-1} - t_j}$

Non-uniform B-splines (NUBS) are the most popular tools for geometric modeling of curves and surfaces, and the non-uniform rational B-spline (NURBS) curves and surfaces, as the extension of NUBS, have become a de facto standard in industry of geometric modeling because they provide a common mathematical form for analytical geometry and data exchange between different CAD systems. Traditionally, almost all B-spline functions and their derivatives are recursively evaluated with de Boor - Cox recursion formula [11, 12], but such a computation, which is actually fulfilled by a recursive procedure implemented with de Boor - Cox recursion formula, is not as efficient as a non-recursive procedure implemented with an explicit formula such as a matrix-form formula or other explicit ones, since the recursive procedure needs more stack operations and computer memories than a non-recursive procedure, besides its computational complexity. In general, the explicit matrix representation of B-splines makes it easier and faster to evaluate B-spline curves and surfaces because the polynomial evaluation is more efficient on a power basis that enables the efficient computation using

Horner' s schema [31].

The matrix-form representations of curves and surfaces are widely applied in CAD systems, since matrix formulations are very compact to write, clear to understand, simple to program and efficient to implement. They facilitate the basis transformation between the common power basis and the B-spline basis functions. The polynomial space spanned by the B-spline basis can be converted into the piecewise polynomial one spanned by the power basis, so that both the matrix representations of non-uniform B-spline curves/surfaces and the explicit computation of de Boor - Cox recursion formula are always possible, but unfortunately, so far no any explicit evaluation scheme for de Boor - Cox recursion formula has been available yet.

A few matrix representations for Bézier, uniform B-spline and NURBS curves and surfaces have been published. The matrix representation for Bézier curves of arbitrary degree was presented in 1982 [7, 10], while the general matrix formula for uniform B-spline curves of arbitrary degree was given in [10]. These two matrix representations for Bézier and uniform B-spline curves are equivalent to the two recursive matrix formulae represented in [28] in the context of the computing results, respectively. Ding and Davies derived the matrix representations of non-uniform B-spline curves up to degree 3 from de Boor - Cox algorithm [14]. A recursive symbolic computation for the matrix representation of NURBS curves and surfaces of arbitrary degree [9] was given by using Böhm' s knot insertion algorithm [3]. The recursive coefficient formula for the elements of the coefficient matrix of non-uniform B-spline curves was presented in [18] by means of de Boor - Cox algorithm, but the recursive formula of the matrix itself other than the elements of the coefficient matrix for non-uniform B-spline curves and surfaces of arbitrary degree was derived in [28] based on de Boor - Cox recursion formula and Toeplitz matrices representing the product of two polynomials. Both of these formulae for the matrix representations of non-uniform B-spline curves and surfaces of arbitrary degree were implemented by the recursive procedures not analytical explicit ones. Explicit matrix representations for NURBS curves and surfaces were derived from the computation of divided difference [24, 27], in which a linear system of equations had to be solved, so that much intensive computing for the $(k+1)\times(k+1)$ determinant and the $k+1$ cofactors was cost. Another explicit matrix formula based on the Marsden identity in [24] costs much intensive computing for the $(k+1)\times(k+1)$ determinant and its $k+1$ cofactors, too.

## 1.2 Box spline

Box spline represents a generalization of univariate spline theory to

several variables. Box splines are introduced by De Boor and DeVore [13]. A comprehensive treatment of box splines and their general theory is given in the book called "box spline book" by De Boor, Höllig and Riemenschneider [4].

An box spline is defined by some k directions $v_i$ in $R^s$. We assume that $k \geq s$ and $v_1, \cdots, v_s$ are linearly independent. Then the box splines $B_k(x) := B(x \mid v_1 ... v_k), k = s+1, ..., k$ , are defined by successive convolutions[5, 17, 29],

$$B_s(x) := \begin{cases} 1/\det[v_1 ... v_s], & if \ \ x \in [v_1 ... v_s][0,1)^s \\ 0 & else \end{cases}$$

$$B_k(x) := \int_0^1 B_{k-1}(x - tv_k)dt, \ \ k > s$$

A particular example of box splines is the b-splines with equidistant knots.

In general, box splines consist of regularly arranged polynomial pieces and they have a useful geometric interpretation. Of particular interest for geometric design are box spline surfaces that consist of triangular polynomial pieces. These box spline surfaces have planar domains, but it is quite simple to construct arbitrary two-dimensional surfaces, i.e., manifolds, with these box splines.

Because a box (the outer product of a set of intervals) can always be subdivided into smaller boxes by subdividing those intervals, we can always express a box-spline basis function over a coarse grid in terms of the sum of basis functions over a nested finer grid.

Box-spline subdivision approach is applied to the 3-direction quartic c2 box-spline by Loop [25]. It is defined over any polyhedron with triangular faces.

## 1.3 Biorthogonal subdivision wavelets based on the lifting scheme

Subdivision-based wavelet construction deals with the wavelet synthesis and analysis processes. The lifting scheme [30, 35, 36] provides an elegant framework for the construction of second generation wavelet transforms. Many lifting based wavelet are constructed as Loop [21, 25, 1], Catmull-Clark [37, 35], Doo-Sabin [41] and $\sqrt{3}$ [39] subdivisions.

Suppose a high-resolution subdivision model $\left[M^{i+1}, V^{i+1}\right]_{Loop}$, where $M^{i}$ and $V^{i+1}$ is the control mesh and a column vector of the control points of the

Loop surface at resolution level i+1 respectively, is decomposed into a lower-resolution model $\left[M^i,V^i\right]_{Loop}$ and a column vector of the wavelet coefficient $W^i$. The wavelet analysis process can be represented as

$$V^i = A\left(V^{i+1}\right)$$

$$W^i = B\left(V^{i+1}\right)$$

The wavelet synthesis is the inverse process of the above wavelet analysis, a process from a lower resolution subdivision model to the higher one (see Figure 1). It can be represented as $V^{i+1} = P\left(V^i\right) + Q\left(W^i\right)$.

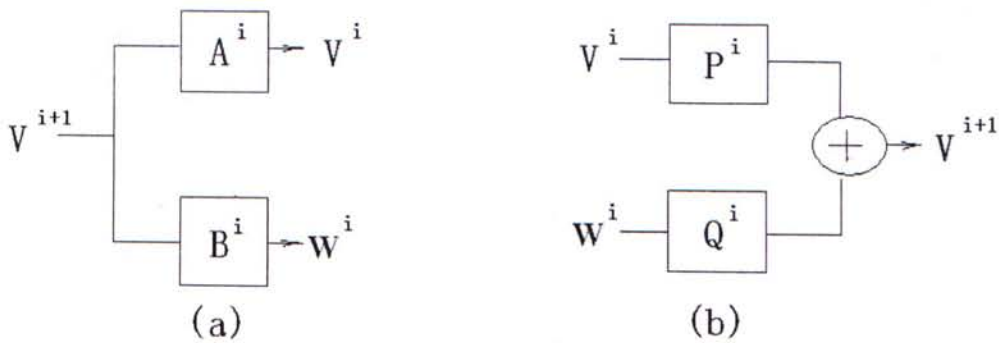



Figure 1. (a) analysis; (b) synthesis.

Based on the lifting scheme, we can construct a new wavelet transform with synthesis filters $(\hat{P}^i, \hat{Q}^i)$, where $\hat{P}^i$ is the same as $P^i$, and $\hat{Q}^i$ can be obtained from $P^i$ and $Q^i$ with the lifting scheme as shown in Figure 2.

$$\hat{Q}^i = Q^i + P^i S^i,$$

where $S^i$ is the lifting matrix.

The synthesis process of subdivision wavelet can be redefined as

$$V^{i+1} = \hat{P}^i\left(V^i\right) + \hat{Q}^i\left(E^i\right) = P^i(V^i + S^i E^i) + Q^i(E^i).$$

Then the wavelet analysis can be represented as:

$$V^i = A^i\left(V^{i+1}\right) - S^i E^i$$

$$E^i = B^i\left(V^{i+1}\right)$$

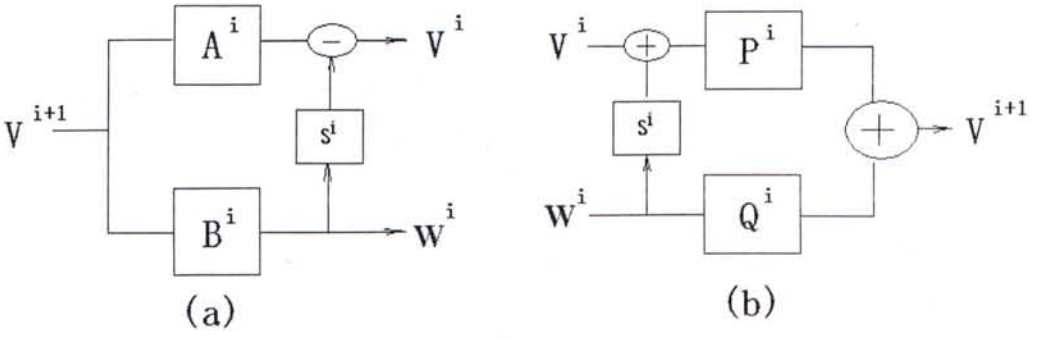



Figure2: Subdivision wavelets with lifting scheme

## 1.4 Geometrically-constrained subdivision wavelets

Geometric modeling with sharp features has been considered not only in B-spline and NURBS surfaces, but also in subdivision-based wavelet transforms. In general, the research of geometrically-constrained modeling of subdivision surfaces concentrates on developing special interpolation subdivision rules and treating sharp features.

The interpolation subdivisions [8, 32, 23] proved highly efficient for subdivision surface modeling under position constraints. Geometric modeling with sharp features can also be considered in subdivision wavelets. In the $\sqrt{3}$ subdivision wavelet construction, boundary constraints were treated [39], while the sharp-feature modeling based on the $\sqrt{2}$ and the Doo-Sabin subdivision wavelet transforms was presented in [41] and [40], respectively. All these works adopted such a common way that only the subdivision rules for preserving sharp features were changed in the multi-resolution geometric modeling. By means of the ability for customizing wavelet constructions of the lifting scheme, Li, Qin and Sun constructed the B-spline wavelet transforms satisfying geometric constraints in curve modeling [22].

Inspired by [22], our subdivision wavelet construction, which will be given in Chapter 3, is based on geometrically constrained lifting operations without any change of the subdivision rules.

## 1.5 Contributions

**Explicit computation and matrix formulations of B-spline curves and surfaces**

None of the previous methods can give us any matrix representation of non-uniform B-spline curves, but my explicit formulae and Matlab programs can produce the matrix formula of non-uniform B-spline and NURBS

curves, such that it is much more efficient and easy for my methods to compute B-spline and NURBS curves and their derivatives than the previous methods. On the other hand, this thesis also gives an explicit formula for a general recursion scheme. It can be used for computing derivatives of non-uniform B-spline and NURBS curves explicitly.

**Biorthogonal subdivision wavelet transforms based on lifting schemes with geometric constraints**

In dual subdivisions like Doo-Sabin scheme, all the old control vertices disappear after one subdivision step. This becomes a big challenge to the biorthogonal subdivision-based wavelet construction. In our biorthogonal wavelet construction based on the dual subdivision, we construct the synthesis and analyesis rules, by using the old vertices. We use the lifting scheme to keep the fitting quality of the wavelet transform, and the local orthogonalization to improve the computing efficiency.

The motivation of our work - subdivision wavelet based on lifting scheme with geometric constraints, is to improve the representation of the complicated models at low resolution levels and facilitate the interactive modeling of geometric models through interpolating some specified constraints. The previous methods need to change the subdivision rules to keep the sharp features. This made the subdivision-based wavelet transform algorithms more complicated and difficult to implement. Furthermore, they cannot deal with high-order derivative constraints (they can deal with only position and 1st - order derivative constraints). In this thesis, I present the novel lifting schemes for preserving various geometric constraints, including position and n-th ($n^3$ 1) order derivative constraints as well as curve constraints. The novel lifting schemes make the algorithms of the subdivision-based wavelet transforms more concise and easy to implement. They do not need to change any subdivision rules. These are the main differences between my methods and other ones.

## 2 Explicit symbol formulae for B-splines

The B-spline curve can be computed by de Boor algorithm [12, 16] as follows:

$$\mathbf{C}(t)=\sum_{j=0}^{n}\mathbf{P}_j N_{j,k}(t)=\sum_{j=i-k}^{i}\mathbf{P}_j N_{j,k}(t) \tag{2.1}$$
$$=\sum_{j=i-k}^{i-1}\mathbf{P}_j^1 N_{j,k-1}(t)=\cdots=\mathbf{P}_{i-k}^k, t_i\le t<t_{i+1}$$

where $N_{j,k}(t)$ are the B-spline basis functions, and

$$\mathbf{P}_j^r=\begin{cases}\mathbf{P}_j, & r=0;\\ \beta_j^r(t)\mathbf{P}_j^{r-1}+\alpha_j^r(t)\mathbf{P}_{j+1}^{r-1}, & r=1,2,\cdots,k;\\ & j=i-k,i-k+1,\cdots,i-r;\end{cases} \tag{2.2}$$

$$\begin{cases}\alpha_j^r(t)=\dfrac{t-t_{j+r}}{t_{j+k+1}-t_{j+r}};\\ \beta_j^r(t)=1-\alpha_j^r(t);\end{cases} \tag{2.3}$$

with the conversion 0/0 = 0.

Note that Equations (2.1)-(2.3) look unlike the original form of de Boor algorithm [12], but they are from [16] and actually identical to the original one.

## 2.1 Explicit formula for a general recursion scheme

In order to derive the explicit computation of de Boor algorithm, an explicit computation formula for the general recursion scheme is given as follows first.

**Theorem 1** Given a set of initial values of scalars or vectors such as 2D or 3D points, $p_{i-k},p_{i-k+1},\cdots,p_i$, the recursion scheme

$$p_j^r=\beta_j^r(t)p_j^{r-1}+\alpha_j^r(t)p_{j+1}^{r-1},\quad r=1,2,\cdots,k, \tag{2.4}$$

with the initial condition

$$p_j^0=p_j, j=i-k,i-k+1,\cdots,i,$$

can be computed explicitly as

$$p_{i-k}^{k}=\begin{bmatrix}p_{i-k} & p_{i-k+1} & \cdots & p_i\end{bmatrix}\begin{bmatrix}\gamma_{i-k}(t)\\ \gamma_{i-k+1}(t)\\ \vdots\\ \gamma_i(t)\end{bmatrix}, \tag{2.5}$$

where

$$\gamma_{i-k+m}(t)=\begin{cases}\prod_{u=1}^{k}\beta_{i-k}^{u}(t), & m=0;\\ \sum_{a_1=m}^{k}\left\{\sum_{a_2=m-1}^{a_1-1}\left\{\sum_{a_3=m-2}^{a_2-1}\{\cdots\right.\right.\\ \times\left\{\sum_{a_m=1}^{a_{m-1}-1}\left\{\left\{f_1^{a_m-1}(t,i-k+m)\right\}\alpha_{i-k+m-1}^{a_m}(t)\right.\right.\\ \left.\left.\times f_{a_m+1}^{a_{m-1}-1}(t,i-k+m-1)\right\}\right\}\cdots\alpha_{i-k+2}^{a_3}(t)f_{a_3+1}^{a_2-1}(t,i-k+2)\right\}\\ \left.\times\alpha_{i-k+1}^{a_2}(t)f_{a_2+1}^{a_1-1}(t,i-k+1)\right\}\\ \left.\times\alpha_{i-k}^{a_1}(t)f_{a_1+1}^{k}(t,i-k)\right\},m=1,\cdots,k;\end{cases} \tag{2.6}$$

$$f_b^c(t,\ell)\overset{\Delta}{=}\prod_{j=b}^{c}\beta_\ell^j(t)=\begin{cases}1 & \text{if } b>c;\\ \beta_\ell^b(t)\beta_\ell^{b+1}(t)\cdots\beta_\ell^c(t) & \text{if } b\le c.\end{cases}$$

$\beta_j^r(t)$ and $\alpha_j^r(t)$ are the recursion coefficients, which can be either constants or variant functions with a variable.

**Proof.** This theorem can be proved by mathematical induction.

When $k=1$, we have

$$\gamma_{i-1+m}(t)=\begin{cases}\beta_{i-1}^{1}(t), & m=0;\\ \alpha_{i-1}^{1}(t), & m=1.\end{cases}$$

Obviously, Equations (2.5) and (2.6) are true.

Suppose that Equations (2.5) and (2.6) are true when $k=k'$, next, we will prove that Equations (2.5) and (2.6) are also true when $k=k'+1$.

From Equation (2.4), we have

$$P_{i-k'-1}^{k'+1}=\beta_{i-k'-1}^{k'+1}(t)P_{i-k'-1}^{k'}+\alpha_{i-k'-1}^{k'+1}(t)P_{i-k'}^{k'}$$

Substituting Equation (2.5) into the right hand side of the above equation yields

$$P_{i-k'-1}^{k'+1} = \beta_{i-k'-1}^{k'+1}(t)\begin{bmatrix} P_{i-k'-1}^{k'} & P_{i-k'}^{k'} & \cdots & P_{i-1}^{k'} \end{bmatrix}\begin{bmatrix} \gamma_{i-k'-1}(t) \\ \gamma_{i-k'}(t) \\ \vdots \\ \gamma_{i-1}(t) \end{bmatrix}$$

$$+\alpha_{i-k'-1}^{k'+1}(t)\begin{bmatrix} P_{i-k'}^{k'} & P_{i-k'+1}^{k'} & \cdots & P_{i}^{k'} \end{bmatrix}\begin{bmatrix} \gamma_{i-k'}(t) \\ \gamma_{i-k'+1}(t) \\ \vdots \\ \gamma_{i}(t) \end{bmatrix}$$

$$=\begin{bmatrix} P_{i-k'-1}^{k'} & P_{i-k'}^{k'} & P_{i-k'+1}^{k'} & \cdots & P_{i}^{k'} \end{bmatrix}\begin{bmatrix} \beta_{i-k'-1}^{k'+1}(t)\gamma_{i-k'-1}(t) \\ \left(\beta_{i-k'-1}^{k'+1}(t)+\alpha_{i-k'-1}^{k'+1}(t)\right)\gamma_{i-k'}(t) \\ \left(\beta_{i-k'-1}^{k'+1}(t)+\alpha_{i-k'-1}^{k'+1}(t)\right)\gamma_{i-k'+1}(t) \\ \vdots \\ \left(\beta_{i-k'-1}^{k'+1}(t)+\alpha_{i-k'-1}^{k'+1}(t)\right)\gamma_{i-1}(t) \\ \alpha_{i-k'-1}^{k'+1}(t)\gamma_{i}(t) \end{bmatrix}.$$

Substituting Equation (2.6) into the above equation yields

i) $\gamma_{i-(k'+1)}(t) = \prod_{a=1}^{(k'+1)} \beta_{i-(k'+1)}^{a}(t)$, if $m$=0;

ii) $\gamma_{i-(k'+1)+m}(t) = \alpha_{i-k'-1}^{k'+1}(t)\gamma_{i}(t)$

$$= \alpha_{i-(k'+1)}^{k'+1}(t)\prod_{a=1}^{k'} \alpha_{i-a}^{a}(t) = \prod_{a=1}^{k'+1} \alpha_{i-a}^{a}(t), \text{ if } m = k'+1;$$

iii) if $1 \le m \le k'$,

$$\gamma_{i-(k'+1)+m}(t) = \left(\beta_{i-k'-1}^{k'+1}(t)+\alpha_{i-k'-1}^{k'+1}(t)\right)\gamma_{i-k'+m-1}(t)$$

$$= \left(\beta_{i-(k'+1)}^{k'+1}(t)+\alpha_{i-(k'+1)}^{k'+1}(t)\right)\gamma_{i-(k'+1)+m}(t)$$

$$= \left(\beta_{i-(k'+1)}^{k'+1}(t) + \alpha_{i-(k'+1)}^{k'+1}(t)\right) \sum_{a_1=m}^{k'} \left\{ \sum_{a_2=m-1}^{a_1-1} \left\{ \sum_{a_3=m-2}^{a_2-1} \{\cdots \right.\right.$$

$$\left\{ \sum_{a_m=1}^{a_{m-1}-1} \left\{ \left\{ f_1^{a_m-1}(t, i-(k'+1)+m) \right\} \alpha_{i-(k'+1)+m-1}^{a_m}(t) \right.\right.$$

$$\left.\left. f_{a_m+1}^{a_{m-1}-1}(t, i-(k'+1)+m-1) \right\} \right\} \cdots$$

$$\alpha_{i-(k'+1)+2}^{a_1}(t) f_{a_3+1}^{a_2-1}(t, i-(k'+1)+2) \Big\} \alpha_{i-(k'+1)+1}^{a_2}(t)$$

$$f_{a_2+1}^{a_1-1}(t, i-(k'+1)+1) \Big\} \alpha_{i-(k'+1)}^{a_1}(t) f_{a_1+1}^{k'}(t, i-(k'+1)) \Big\}$$

$$= \sum_{a_1=m}^{k'+1} \left\{ \sum_{a_2=m-1}^{a_1-1} \left\{ \sum_{a_3=m-2}^{a_2-1} \left\{ \cdots \left\{ \sum_{a_m=1}^{a_{m-1}-1} \left\{ \left\{ f_1^{a_m-1}(t, i-(k'+1)+m) \right\} \right.\right.\right.\right.$$

$$\left.\left.\left.\left. \alpha_{i-(k'+1)+m-1}^{a_m}(t) f_{a_m+1}^{a_{m-1}-1}(t, i-(k'+1)+m-1) \right\} \right\} \cdots \alpha_{i-(k'+1)+2}^{a_3}(t) f_{a_3+1}^{a_2-1}(t, i-(k'+1)+2) \right\}$$

$$\alpha_{i-(k'+1)+1}^{a_2}(t) f_{a_2+1}^{a_1-1}(t, i-(k'+1)+1) \Big\}$$

$$\alpha_{i-(k'+1)}^{a_1}(t) f_{a_1+1}^{k'+1}(t, i-(k'+1)) \Big\}, m = 1, \cdots, k'+1.$$

In the above rightmost derivation, note that

$$f_{k'+2}^{k'}(t, i-(k'+1)) = f_{k'+2}^{k'+1}(t,\ i-(k'+1)) = 1.$$

Thus, it is clear that actually this theorem is always true.

## 2.2 Explicit formulae for de Boor algorithms of B-spline curves and their derivatives

### 2.2.1 Explicit computation of de Boor Algorithm for Computing B-Spline Curves

**Theorem 2** The non-uniform B-spline curve $C(t)$, $t_i \le t < t_{i+1}$, can explicitly be computed by de Boor algorithm for B-spline curves as follows:

$$\mathbf{C}(t) = \mathbf{P}_{i-k}^{k} = \begin{bmatrix} \mathbf{P}_{i-k} & \mathbf{P}_{i-k+1} & \cdots & \mathbf{P}_i \end{bmatrix} \begin{bmatrix} \gamma_{i-k}(t) \\ \gamma_{i-k+1}(t) \\ \vdots \\ \gamma_i(t) \end{bmatrix}, t_i \le t < t_{i+1}. \qquad (2.7)$$

The B-spline basis functions of degree $k$ can also be computed explicitly as below:

$$N_{i-k+m,k}(t)=\begin{cases}\prod_{a=1}^{k}\beta_{i-k}^{a}(t), & \text{if } m=0;\\ \sum_{a_1=m}^{k}\left\{\sum_{a_2=m-1}^{a_1-1}\left\{\sum_{a_3=m-2}^{a_2-1}\left\{\cdots\left\{\sum_{a_m=1}^{a_{m-1}-1}\left\{\left\{f_1^{a_m-1}(t,i-k+m)\right\}\right.\right.\right.\right.\\ \quad\left.\left.\left.\left.\times\alpha_{i-k+m-1}^{a_m}(t)f_{a_m+1}^{a_{m-1}-1}(t,i-k+m-1)\right\}\right\}\cdots\right.\\ \quad\left.\times\alpha_{i-k+2}^{a_3}(t)f_{a_3+1}^{a_2-1}(t,i-k+2)\right\}\\ \quad\left.\times\alpha_{i-k+1}^{a_2}(t)f_{a_2+1}^{a_1-1}(t,i-k+1)\right\}\\ \quad\left.\times\alpha_{i-k}^{a_1}(t)f_{a_1+1}^{k}(t,i-k)\right\}, & \text{if } m=1,\cdots,k;\end{cases}\tag{2.8}$$

**Proof.** Comparing the right hand sides of Equations (2.1) and (2.7) as well as substituting Equation (2.3) for $\beta_j^r(t)$ and $\alpha_j^r(t)$ in Equation (2.4) yields Equation (2.8). ‖

Equations (2.6) and (2.8) cannot be gotten by a straightforward textual substitution of the recursion formula presented by de Boor-Cox. Actually, they are the explicit representations of B-splines of arbitrary degree. The Matlab programs for the explicit symbolical matrix formula of B-spline curves of arbitrary degree are given in the Appendix, using Equation (2.8).

## 2.2.2 Explicit computation of Derivatives of B-Spline Curves

The de Boor algorithm can be generalized to compute derivatives of B-spline curves [12, 19], and the $\ell$-th derivative of the B-spline curve is computed [16] as follows:

$$\mathbf{C}^{(\ell)}(t)=\frac{d}{dt^{\ell}}\sum_{j=0}^{n}\mathbf{P}_jN_{j,k}(t)=\sum_{j=i-k}^{i-1}\mathbf{P}_j^1N_{j,k-\ell}(t)=\cdots=\sum_{j=i-k}^{i-\ell}\mathbf{P}_j^{\ell}N_{j,k-\ell}(t),$$
$$t_i\le t<t_{i+1}\text{ and }0\le\ell\le k,$$

where

$$\mathbf{P}_j^r=\begin{cases}\mathbf{P}_j, & r=0;\\ \overline{\beta}_j^r\mathbf{P}_j^{r-1}+\overline{\alpha}_j^r\mathbf{P}_{j+1}^{r-1}, & r=1,2,\cdots,\ell;\\ & j=i-k,i-k+1,\cdots,i-\ell;\end{cases}$$

$$\bar{\alpha}_j^r = \frac{k-r+1}{t_{j+k+1} - t_{j+r}};$$

$$\bar{\beta}_j^r = -\bar{\alpha}_j^r.$$

Note that the above derivative formula looks unlike the original form in [12], but actually they are identical to that, deduced from the derivative formula in the Section 8.9 of [16].

Similarly to Section 2.1, we can prove the following theorem with the principle of mathematical induction:

**Theorem 3** The $\ell$-th derivative of the B-spline curve can be computed explicitly as follows:

$$C^{(\ell)}(t) = \begin{bmatrix} \mathbf{P}_{i-k}^{\ell} & \mathbf{P}_{i-k+1}^{\ell} & \cdots & \mathbf{P}_{i-\ell}^{\ell} \end{bmatrix} \begin{bmatrix} N_{i-k,k-\ell}(t) \\ N_{i-k+1,k-\ell}(t) \\ \vdots \\ N_{i-\ell,k-\ell}(t) \end{bmatrix}$$

where

$$\mathbf{P}_{i-k+j}^{\ell} = \begin{bmatrix} \mathbf{P}_{i-k+j} & \mathbf{P}_{i-k+1+j} & \cdots & \mathbf{P}_{i-k+\ell+j} \end{bmatrix} \begin{bmatrix} \bar{\gamma}_{i-k+j} \\ \bar{\gamma}_{i-k+1+j} \\ \vdots \\ \bar{\gamma}_{i-k+\ell+j} \end{bmatrix},$$

$$j = 0,1,\cdots,k-\ell;$$

$$\bar{\gamma}_{i-k+j+m} = \begin{cases} \prod_{a=1}^{\ell} \bar{\beta}_{i-k+j}^{a}(t) & \text{if } m = 0, \\ \sum_{a_1=m}^{\ell} \left\{ \sum_{a_2=m-1}^{a_1-1} \left\{ \sum_{a_3=m-2}^{a_2-1} \left\{ \cdots \left\{ \sum_{a_m=1}^{a_{m-1}-1} \left\{ \left\{ g_1^{a_m-1}(i-k+j+m) \right\} \right. \right. \right. \right. \\ \quad \left. \left. \times \bar{\alpha}_{i-k+j+m-1}^{a_m} g_{a_m+1}^{a_{m-1}-1}(i-k+j+m-1) \right\} \right\} \cdots \\ \quad \times \bar{\alpha}_{i-k+j+2}^{a_3} g_{a_3+1}^{a_2-1}(i-k+j+2) \Big\} \\ \quad \times \bar{\alpha}_{i-k+j+1}^{a_2} g_{a_2+1}^{a_1-1}(i-k+j+1) \Big\} \\ \quad \times \bar{\alpha}_{i-k+j}^{a_1} g_{a_1+1}^{\ell}(i-k+j) \Big\} & \text{if } m = 1,\cdots,\ell, \end{cases}$$

$$j = 0,1,\cdots,k-\ell; \qquad (2.9)$$

$$g_b^c(i) \overset{\Delta}{=} \prod_{j=b}^{c} \bar{\beta}_i^j = \begin{cases} 1, & b > c; \\ \bar{\beta}_i^b \bar{\beta}_i^{b+1} \cdots \bar{\beta}_i^c, & b \le c. \end{cases} \quad \|$$

Actually Equation (2.6) can be regarded as a special case of Equation

(2.9) when $\ell = k$.

In the next section, we will derive an explicit power-basis matrix representation for non-uniform B-spline curves from the explicit formula for de Boor algorithm of B-spline curves.

## 2.3 Explicit power-basis matrix fomula for non-uniform B-spline curves

Theorem 4 The non-uniform B-spline curve of degree $k$

$$\mathbf{C}(t) = \sum_{j=0}^{n} \mathbf{P}_j N_{j,k}(t) = \sum_{j=i-k}^{i} \mathbf{P}_j N_{j,k}(t), \quad t_i \le t < t_{i+1}$$

can be represented by an explicit power-basis matrix form as

$$C(t) = \begin{bmatrix} 1 \\ t \\ \vdots \\ t^r \\ \vdots \\ t^k \end{bmatrix}^T \begin{bmatrix} 1 & \frac{C_1^0 t_i^1}{(t_i - t_{i+1})^1} & \cdots & \frac{C_r^0 t_i^r}{(t_i - t_{i+1})^r} & \cdots & \frac{C_k^0 t_i^k}{(t_i - t_{i+1})^k} \\ \frac{(-1)^1 C_1^1 t_i^0}{(t_i - t_{i+1})^1} & \cdots & \frac{-C_r^1 t_i^{r-1}}{(t_i - t_{i+1})^r} & \cdots & \frac{-C_k^1 t_i^{k-1}}{(t_i - t_{i+1})^k} \\ & \ddots & \vdots & \vdots & \vdots \\ & & \frac{(-1)^r C_r^r t_i^0}{(t_i - t_{i+1})^r} & \vdots & \vdots \\ & & & \ddots & \vdots \\ 0 & & & & \frac{(-1)^k C_k^k t_i^0}{(t_i - t_{i+1})^k} \end{bmatrix} \begin{bmatrix} A_{0,0}^k & A_{0,1}^k & \cdots & A_{0,k}^k \\ A_{1,0}^k & A_{1,1}^k & \cdots & A_{1,k}^k \\ \vdots & \vdots & \cdots & \vdots \\ A_{k,0}^k & A_{k,1}^k & \cdots & A_{k,k}^k \end{bmatrix} \begin{bmatrix} \mathbf{P}_{i-k} \\ \mathbf{P}_{i-k+1} \\ \vdots \\ \mathbf{P}_i \end{bmatrix},$$

$$t \in [t_i, t_{i+1}), \quad (2.10)$$

with the conversion 0/0 = 0 again, where

$$C_r^j = \frac{r!}{j!(r-j)!};$$

$$A_{0,m}^{k}=\begin{cases}\prod_{a=1}^{k}\beta_{i-k}^{a}(0) \quad \text{if } m=0,\\ \sum_{a_1=m}^{k}\left\{\sum_{a_2=m-1}^{a_1-1}\left\{\sum_{a_3=m-2}^{a_2-1}\left\{\cdots\left\{\sum_{a_m=1}^{a_{m-1}-1}\left\{\left\{f_1^{a_m-1}(0,i-k+m)\right\}\right.\right.\right.\right.\\ \quad\left.\left.\times\alpha_{i-k+m-1}^{a_m}(0)f_{a_m+1}^{a_{m-1}-1}(0,i-k+m-1)\right\}\right\}\cdots\\ \quad\times\alpha_{i-k+2}^{a_3}(0)f_{a_3+1}^{a_2-1}(0,i-k+2)\Big\}\\ \quad\times\alpha_{i-k+1}^{a_2}(0)f_{a_2+1}^{a_1-1}(0,i-k+1)\Big\}\\ \quad\times\alpha_{i-k}^{a_1}(0)f_{a_1+1}^{k}(0,i-k)\Big\}\ \text{if } m=1,\cdots,k;\end{cases}\tag{2.11}$$

$$A_{j,m}^{k}=$$

$$\begin{cases}\frac{1}{j!}\sum\limits_{\substack{b_1+b_2+\cdots+b_k=j\\0\le b_1,b_2,\cdots,b_k\le j}}\frac{j!}{b_1!b_2!\cdots b_k!}\beta_{i-k}^{1\,(b_1)}(0)\beta_{i-k}^{2\,(b_2)}(0)\cdots\beta_{i-k}^{k\,(b_k)}(0)\ \text{if } m=0,\\ \frac{1}{j!}\sum\limits_{a_1=m}^{k}\left\{\sum\limits_{\substack{b_{1,1}+b_{1,2}+b_{1,3}=j\\0\le b_{1,1},b_{1,2},b_{1,3}\le m}}\frac{j!}{b_{1,1}!b_{1,2}!b_{1,3}!}\left\{\sum\limits_{a_2=m-1}^{a_1-1}\sum\limits_{\substack{b_{2,1}+b_{2,2}+b_{2,3}=b_{1,1}\\0\le b_{2,1},b_{2,2},b_{2,3}\le b_{1,1}}}\frac{b_{1,1}!}{b_{2,1}!b_{2,2}!b_{2,3}!}\right.\right.\\ \times\left\{\sum\limits_{a_3=m-2}^{a_2-1}\sum\limits_{\substack{b_{3,1}+b_{3,2}+b_{3,3}=b_{2,1}\\0\le b_{3,1},b_{3,2},b_{3,3}\le b_{1,1}}}\frac{b_{2,1}!}{b_{3,1}!b_{3,2}!b_{3,3}!}\left\{\cdots\left\{\sum\limits_{a_m=1}^{a_{m-1}-1}\right.\right.\right.\\ \times\left\{\sum\limits_{\substack{b_{m,1}+b_{m,2}+b_{m,3}=b_{m-1,1}\\0\le b_{m,1},b_{m,2},b_{m,3}\le b_{m-1,1}}}\frac{b_{m-1,1}!}{b_{m,1}!b_{m,2}!b_{m,3}!}\left\{f_1^{a_m-1^{(b_{m,1})}}(0,i-k+m)\right\}\right.\\ \left.\left.\times\alpha_{i-k+m-1}^{a_m\,(b_{m,2})}(0)f_{a_m+1}^{a_{m-1}-1^{(b_{m,3})}}(0,i-k+m-1)\right\}\right\}\cdots\Big\}\\ \times\alpha_{i-k+2}^{a_3\,(b_{3,2})}(0)f_{a_3+1}^{a_2-1^{(b_{3,3})}}(0,i-k+2)\Big\}\\ \times\alpha_{i-k+1}^{a_2\,(b_{2,2})}(0)f_{a_2+1}^{a_1-1^{(b_{2,3})}}(0,i-k+1)\Big\}\\ \times\alpha_{i-k}^{a_1\,(b_{1,2})}(0)f_{a_1+1}^{k\,(b_{1,3})}(0,i-k)\Big\}\quad\text{if } m=1,\cdots,k,\end{cases}$$

$$j=1,\ 2,\ \cdots,\ k;\tag{2.12}$$

and the superscript, $(\xi)$, denotes the $\xi$-th derivative with respect to $u=(t-t_i)/(t_{i-1}-t_i)$, which is the parameter of the B-spline curve equation substituting for the original parameter $t$ in the equation of the curve;

$$f_b^{c^{(\xi)}}(0,j)=\frac{d}{du^{\xi}}f_b^c(u,j)\Big|_{u=0}$$

$$=\begin{cases}\begin{cases}1 & \text{if } \xi=0,\\ 0 & \text{if } \xi\geq 1,\end{cases} & \text{when } b>c;\\ \beta_j^b(0)\beta_j^{b+1}(0)\cdots\beta_j^c(0) \quad \text{if } \xi=0, & \\ \sum\limits_{\substack{\zeta_0+\zeta_1+\cdots+\zeta_a=\xi\\ 0\leq\zeta_0,\zeta_1,\cdots,\zeta_a\leq\xi}} \dfrac{\xi!\beta_j^{b^{(\zeta_0)}}(0)\beta_j^{b+1^{(\zeta_1)}}(0)\cdots\beta_j^{c^{(\zeta_a)}}(0)}{\zeta_0!\zeta_1!\cdots\zeta_a!}, & \text{where } a=c-b,\\ \text{if } \xi\geq 1, \text{ when } b\leq c; & \end{cases}$$

$$\alpha_j^{r^{(\xi)}}(0)=\frac{d}{du^{\xi}}\left\{\frac{t_i-t_{j+r}+u(t_{i+1}-t_i)}{t_{j+k+1}-t_{j+r}}\right\}\Big|_{u=0}$$

$$=\begin{cases}(t_i-t_{j+r})/(t_{j+k+1}-t_{j+r}), \text{ if } \xi=0;\\ (t_{i+1}-t_i)/(t_{j+k+1}-t_{j+r}), \text{ if } \xi=1;\\ 0, \qquad\qquad \text{if } \xi\geq 2;\end{cases}$$

$$\beta_j^{r^{(\xi)}}(0)=\frac{d}{du^{\xi}}\left\{1-\frac{t_i-t_{j+r}+u(t_{i+1}-t_i)}{t_{j+k+1}-t_{j+r}}\right\}\Big|_{u=0}$$

$$=\begin{cases}(t_{j+k+1}-t_i)/(t_{j+k+1}-t_{j+r}), \text{if } \xi=0;\\ -(t_{i+1}-t_i)/(t_{j+k+1}-t_{j+r}), \text{ if } \xi=1;\\ 0, \qquad\qquad \text{if } \xi\geq 2.\end{cases}$$

Note that the factorial $j!=j\times(j-1)\times\cdots\times 1$, and the convention $0!=1$.

**Proof.** Substituting $t=t_i+u(t_{i+1}-t_i)$ into Equations (2.3), (2.6) and (2.8) yields

$$N_{i-k+m,k}(u)=\gamma_{i-k+m}(u), u\in[0,1), \quad m=0,1,\cdots,k; \quad (2.13)$$

And

$$\begin{cases}\alpha_j^r(u)=\dfrac{t_i-t_{j+r}+u(t_{i+1}-t_i)}{t_{j+k+1}-t_{j+r}}, & r=1,2,\cdots,k; j=i-1,i-2,\cdots,i-k.\\ \beta_j^r(u)=1-\alpha_j^r(u), & \end{cases}$$

On the other hand, The B-spline basis function can also be represented as the following form

$$N_{i-k+m,k}(u)=\begin{bmatrix}1 & u & u^2 & \cdots & u^k\end{bmatrix}\begin{bmatrix}A_{0,m}^k\\A_{1,m}^k\\A_{2,m}^k\\\vdots\\A_{k,m}^k\end{bmatrix}. \qquad (2.14)$$

Comparison with the coefficients of the terms with the same power of $u$ on the right hand sides of Equations (2.13) and (2.14) gives

$$A_{j,m}^k=\begin{cases}\dfrac{1}{j!}\dfrac{d}{du^j}\displaystyle\prod_{a=1}^{k}\beta_{i-k}^a(0) & \text{if } m=0,\\ \dfrac{1}{j!}\dfrac{d}{du^j}\Bigg\{\displaystyle\sum_{a_1=m}^{k}\Bigg\{\sum_{a_2=m-1}^{a_1-1}\Bigg\{\sum_{a_3=m-2}^{a_2-1}\{\cdots\\ \times\Bigg\{\displaystyle\sum_{a_m=1}^{a_{m-1}-1}\Big\{\Big\{f_1^{a_m-1}(u,i-k+m)\Big\}\\ \times\alpha_{i-k+m-1}^{a_m}(u)f_{a_m+1}^{a_{m-1}-1}(u,i-k+m-1)\Big\}\Big\}\cdots\\ \times\alpha_{i-k+2}^{a_3}(u)f_{a_3+1}^{a_2-1}(u,i-k+2)\Big\}\alpha_{i-k+1}^{a_2}(u)f_{a_2+1}^{a_1-1}(u,i-k+1)\Big\}\\ \times\alpha_{i-k}^{a_1}(u)f_{a_1+1}^{k}(u,i-k)\Big\}\Big\}\Big|_{u=0} & \text{if } m=1,\cdots,k, \text{ and}\\ & j=0,1,\cdots k.\end{cases}$$

Subsequently, Equations (2.11) and (2.12) follow from the above equation.

Thus, the degree $k$ B-spline curve can be represented by the matrix representation as

$$C(u)=\begin{bmatrix}1 & u & u^2 & \cdots & u^k\end{bmatrix}\begin{bmatrix}\mathbf{A}^k\end{bmatrix}\begin{bmatrix}\mathbf{P}_{i-k}\\\mathbf{P}_{i-k+1}\\\vdots\\\mathbf{P}_i\end{bmatrix}, u\in[0,1), \quad (2.15)$$

Where

$$\begin{bmatrix}\mathbf{A}^k\end{bmatrix}=\begin{bmatrix}A_{0,0}^k & A_{0,1}^k & \cdots & A_{0,k}^k\\A_{1,0}^k & A_{1,1}^k & \cdots & A_{1,k}^k\\\vdots & \vdots & \cdots & \vdots\\A_{k,0}^k & A_{k,1}^k & \cdots & A_{k,k}^k\end{bmatrix}.$$

Since

$$u^r = \frac{1}{(t_i - t_{i+1})^r} \sum_{j=0}^{r} (-1)^j C_r^j t_i^{r-j} t^j$$

$$= \frac{1}{(t_i - t_{i+1})^r} \begin{bmatrix} 1 & t & t^2 & \cdots & t^r \end{bmatrix} \begin{bmatrix} C_r^0 t_i^r \\ -C_r^1 t_i^{r-1} \\ C_r^2 t_i^{r-2} \\ \vdots \\ (-1)^r C_r^r t_i^0 \end{bmatrix}, \qquad (2.16)$$

thus, substituting Equation (16) into Equation (15) yields Equation (2.10). ‖

Several examples of $[\mathrm{A}^k]$ for B-spline curves of degree $k \leqslant 3$ are evaluated by Equations (2.11) and (2.12) as follows:

$$\left[\mathbf{A}^1\right] = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix},$$

$$\left[\mathbf{A}^2\right] = \begin{bmatrix} \dfrac{t_{i+1} - t_i}{t_{i+1} - t_{i-1}} & \dfrac{t_i - t_{i-1}}{t_{i+1} - t_{i-1}} & 0 \\ \dfrac{-2(t_{i+1} - t_i)}{t_{i+1} - t_{i-1}} & \dfrac{2(t_{i+1} - t_i)}{t_{i+1} - t_{i-1}} & 0 \\ \dfrac{t_{i+1} - t_i}{t_{i+1} - t_{i-1}} & (t_i - t_{i+1})\left(\dfrac{1}{t_{i+1} - t_{i-1}} + \dfrac{1}{t_{i+2} - t_i}\right) & \dfrac{t_{i+1} - t_i}{t_{i+2} - t_i} \end{bmatrix},$$

$$\left[\mathbf{A}^3\right] = \begin{bmatrix} A_{0,0}^3 & 1 - A_{0,0}^3 - A_{0,2}^3 & \dfrac{(t_i - t_{i-1})^2}{(t_{i+2} - t_{i-1})(t_{i+1} - t_{i-1})} & 0 \\ -3A_{0,0}^3 & 3A_{0,0}^3 - A_{1,2}^3 & \dfrac{3(t_{i+1} - t_i)(t_i - t_{i-1})}{(t_{i+2} - t_{i-1})(t_{i+1} - t_{i-1})} & 0 \\ 3A_{0,0}^3 & -3A_{0,0}^3 - A_{2,2}^3 & \dfrac{3(t_{i+1} - t_i)^2}{(t_{i+2} - t_{i-1})(t_{i+1} - t_{i-1})} & 0 \\ -A_{0,0}^3 & A_{0,0}^3 - A_{3,2}^3 - A_{3,3}^3 & A_{3,2}^3 & A_{3,3}^3 \end{bmatrix},$$

Where

$$A_{0,0}^3 = \frac{(t_{i+1} - t_i)^2}{(t_{i+1} - t_{i-1})(t_{i+1} - t_{i-2})},$$

$$A_{3,2}^3 = \frac{-A_{2,2}^3}{3} - A_{3,3}^3 - \frac{(t_{i+1} - t_i)^2}{(t_{i+2} - t_i)(t_{i+2} - t_{i-1})}, \quad \text{and}$$

$$A_{3,3}^3 = \frac{(t_{i+1} - t_i)^2}{(t_{i+3} - t_i)(t_{i+2} - t_i)}.$$

Actually, the above explicit symbolical matrix formulae $[\mathrm{A}^k]$, $k$=1, 2, 3,

are obtained by the Matlab programs in the Appendix, which are the Matlab-programming-language implementations of Equations (2.11), (2.12) and (15). The Matlab function *MatrixA(k, i)* in the Appendix A is the main function for the symbolical computations of the matrices of degree $k$ B-spline curves. Note that it has two input parameters, $k$ and $i$, where $k$ denotes the degree of the B-splines, and $i$ should satisfy the following relationships:

$t_i \le t < t_{i+1}$, and

$i \ge k$.

In order to obtain $[A^3]$ as shown in Equation (2.17), for example, first, let $k$=3 and $i$=3, then call the function *MatrixA(k, i)* to obtain the matrix formula of $[A^3]$ when $t_3 \le t < t_4$ ; finally, replace the subscript of every entry of the matrix (suppose it is equal to $j$) by $j-3+i$ to obtain the general matrix formula of $[A^3]$ when $t_i \le t < t_{i+1}$ and $i$ is an arbitrary integer. Similarly, we can obtain the symbolical matrix formula of arbitrary degree B-spline curves by using the Matlab programs.

Since it is very large and complicated, $[A^4]$ is moved from here to Appendix B.

# 3 Biorthogonal subdivision wavelets with geometric constraints

## 3.1 Primal subdivision and dual subdivision

Subdivision schemes can be classified as primal and dual ones according to the different topological rules. Many primal-subdivision-based wavelets over both triangular (e.g., Loop [21][1], $\sqrt{3}$[39]) and quadrilateral (Catmull-Clark[37], $\sqrt{2}$[40]) meshes are proposed and used in animation and engineering applications, but wavelet constructions based on dual subdivision schemes like Doo-Sabin subdivision have not been widely used.

The subdivision rules of primal schemes like Loop and Catmull-Clark subdivisions can be divided into two parts: the computation rules of new inserted vertices and the relocation rules of old control points. However in dual subdivision schemes, all the old control points disappear after one subdivision step and the refined control meshes are constructed only by the new inserted vertices. Figure 3 shows the different topological refinement rules between primal and dual subdivision schemes.

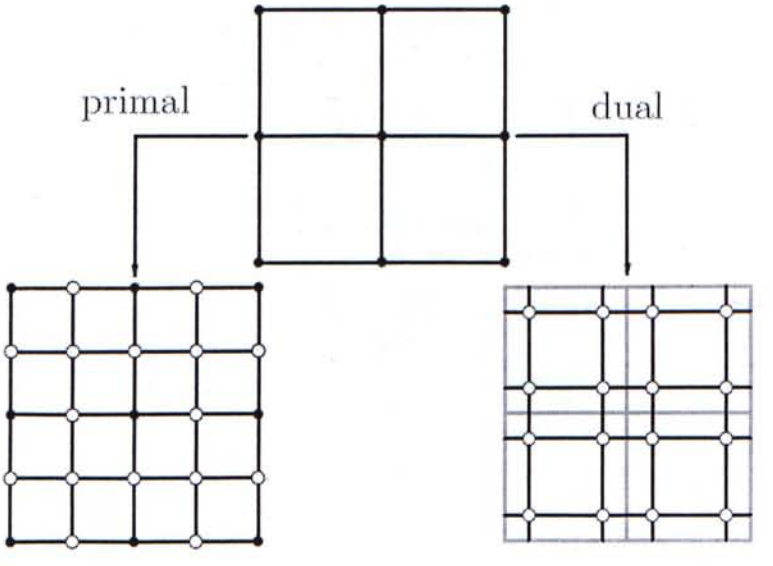



Figure 3: The primal subdivision is a face-split scheme, while the dual subdivision is based on the vertex split. Each face of the base mesh (top) is split into four surfaces by Catmull-Clark subdivision(left), while each vertex is split into four new vertices by DOO-Sabin subdivision (right).

## 3.2 Biorthogonal Loop-subdivision-based wavelets with geometric constraints for triangular meshes

### 3.2.1 Loop subdivision surfaces and exact evaluation

Loop subdivision [25] is proposed based on a quartic box-spline of six direction vectors. The basic rule of Loop-subdivision is to introduce a new vertex, e', for every edge of the mesh and relocate each old vertex v to v'; and then every triangle is split into four smaller ones, introducing new edges. We express the local operators as ones introduced by Li et al. in 2004 [21] (see Figure 4)

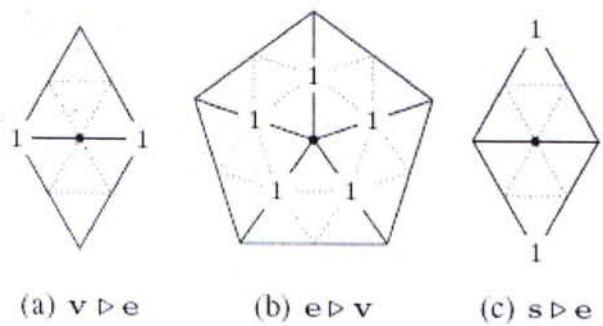



Figure 4: Mask of local operators for Loop subdivision: The operator v ▷ e means "take the average of the two vertex points adjacent to an edge point"; the operator e ▷ v means "take the average of the edge points adjacent to an vertex point"; s ▷ e means "take the average of the two sidewing points adjacent to an edge point";

The procedures of Loop-subdivision are:

$$\begin{cases} e = \frac{3}{4}(v \triangleright e) + \frac{1}{4}(s \triangleright e) \\ v' = \alpha(n)v + (1-\alpha(n))(e \triangleright v) \end{cases} \tag{3.1}$$

where $n$ is the valence of v, and

$$\alpha(n) = \frac{3}{8} + \left(\frac{3}{8} + \frac{1}{4}\cos\frac{2\pi}{n}\right)^2.$$

### 3.2.2 Lifting-based Loop subdivision wavelets

Loop subdivision wavelets were presented individually by Bertram [1] and Li et al. [21] in 2004, respectively. For the sake of completeness,

Lifting-based Loop subdivision wavelets are introduced in this subsection, and then biorthogonal Loop-subdivision wavelets with geometric constraints will be presented in the next subsection.

First of all, Equation (3.1) can be redefined with lifting operations, and its inverse can easily be obtained as follows:

$$\begin{cases} e' = e + \frac{3}{4}(v \triangleright e) + \frac{1}{4}(s \triangleright e) \\ v' = \beta(n)v + (1-\beta(n))(e' \triangleright v) \end{cases} \tag{3.2}$$

where $\beta(n) = \frac{8}{5}\alpha(n) - \frac{3}{5}$.

When the wavelet coefficient vector $e$, is equal to zero, Equation (3.2) provides the same result as Equation (3.1), but the advantage is that the inverse of (3.2) can easily be obtained by moving $v$ and $e$ to the left-hand sides of the equations, respectively, but others to the right-hand sides.

Equation (3.2) can also be regarded as the lazy wavelet synthesis, the wavelet coefficient $e$, may have non-zero value, and the corresponding lazy wavelet basis function, $\psi_\ell$, is already defined by the subdivision process. But the weakness of this wavelet transform is that the fitting quality of the reversing subdivision is very poor.

To overcome this problem, we construct a wavelet basis function $\psi$

$$\psi = \psi_\ell + \sum_{i=0}^{3} \omega_i \phi_i$$

located at $e$ as a linear combination of the related scaling basis functions and the lazy wavelet basis function.

The four coefficients $\omega_i$ can be obtained by orthogonalizing $\psi$ with the four scaling functions $\phi_i$ corresponding to $v_i$ ($i$=0, 1, 2, 3), satisfying

$\langle \psi, \phi_i \rangle = 0$, i=0, 1, 2, 3.

The simplest way to determine the coefficients $\omega_i$ is using the discrete inner product operations [1] rather than accounting the continuous inner products of the basis functions. Using the discrete inner product can simplify the computation of the orthogonalization process and provide a stable fitting operator.

By the discrete orthogonalization, weights $\omega_i$ are computed,

$$<\psi_\ell,\phi_0>=\alpha_0\delta_0+\gamma_1\delta_1+\frac{3}{8}$$

$$<\psi_\ell,\phi_2>=\gamma_0\delta_0+\gamma_1\delta_1+\frac{1}{8}$$

$$<\phi_0,\phi_0>=\alpha_0^2+\gamma_1^2+\gamma_2^2+\gamma_3^2+\frac{n_0-3}{256}+n_0\frac{10}{64}$$

$$<\phi_0,\phi_1>=\alpha_0\gamma_0+\alpha_1\gamma_1+\gamma_2^2+\gamma_3^2+\frac{21}{64}$$

where $\alpha_i=\alpha(n_i),\quad \beta_i=\beta(n_i),$

$\gamma_i=\dfrac{1-\alpha_i}{n_i},\quad \delta_i=\dfrac{1-\beta_i}{n_i}.$

Thus the lifting scheme of Loop-subdivision wavelet transform can be described as follows:

The synthesis process:

$$\mathrm{v}_i \leftarrow \mathrm{v}_i+\omega_i \mathrm{e} \qquad \forall \mathrm{e}, \forall i=0,\ldots,3;$$

$$\mathrm{e} \leftarrow \mathrm{e}+\frac{3}{8}(\mathrm{v}_0+\mathrm{v}_1)+\frac{1}{8}(\mathrm{v}_2+\mathrm{v}_3) \qquad \forall \mathrm{e};$$

$$\mathrm{v} \leftarrow \beta(n)\,\mathrm{v} \qquad \forall \mathrm{v};$$

$$\mathrm{v}_i \leftarrow \mathrm{v}_i+\delta_i \mathrm{e} \qquad \forall \mathrm{e}, \forall i=0,1;$$

The analysis process:

$$\mathrm{v}_i \leftarrow \mathrm{v}_i-\delta_i \mathrm{e} \qquad \forall \mathrm{e}, \forall i=0,1;$$

$$\mathrm{v} \leftarrow \frac{1}{\beta(n)}\mathrm{v} \qquad \forall \mathrm{v};$$

$$\mathrm{e} \leftarrow \mathrm{e}-\frac{3}{8}(\mathrm{v}_0+\mathrm{v}_1)-\frac{1}{8}(\mathrm{v}_2+\mathrm{v}_3) \qquad \forall \mathrm{e};$$

$$\mathrm{v}_i \leftarrow \mathrm{v}_i-\omega_i \mathrm{e} \qquad \forall \mathrm{e}, \forall i=0,\ldots,3;$$

where $\delta_i=\dfrac{1-\beta(n_i)}{n_i}$, and $n_i$ is the valence of $\mathbf{v}_i$.

### 3.2.3 Biorthogonal Loop-subdivision wavelets with geometric constraints

The gray triangle in Figure 5 is the triangular patch containing constraints at the higher level. Let $V^{n}$ be the set of control vertices at a higher level of resolution, which can define that gray triangle. In the above section we have discussed the lifting procedure of wavelets, we analyze $V^{k}$ to get a control vertex set $V^{k-1}$ at one lower resolution level than $V^{k}$ and the corresponding wavelet coefficients $W^{k-1}$. If we subdivide $V^{k-1}$

directly, we can get a control vertex set $V^i$, which is at the same subdivision level as $V^{ii}$ but at lower resolution level. Figure 5 shows a mesh with the control vertex set around an extraordinary vertex.

The size of control vertex set $V^{ii}$ is $K=n+6$, where $n$ is the valence of the vertex at which the origin of the local coordinate system in the triangle is, e. g., the red point in Figure 5). The triangular mesh area of the control vertex set $V^{ii}$ is indicated by $area(H)$.

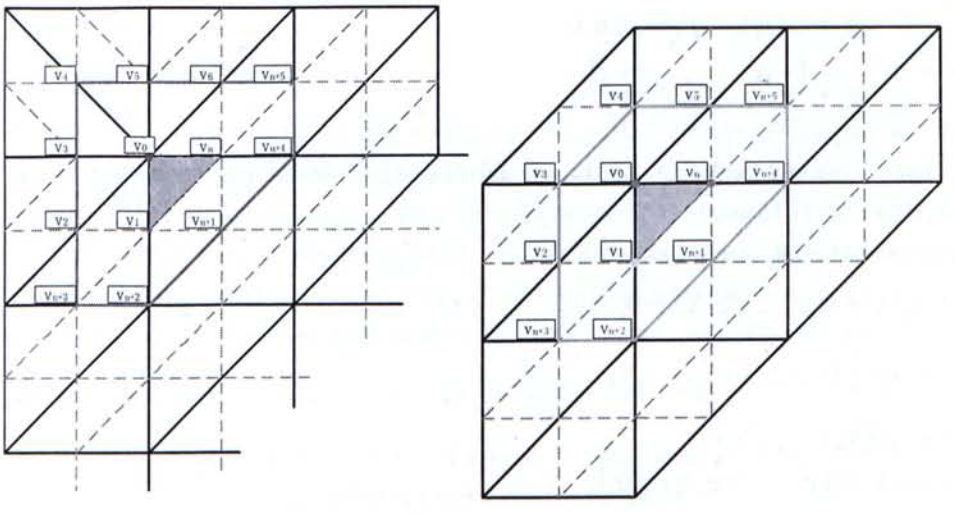


Figure 5: Control vertices of Loop subdivision; the red point(left) is an extraordinary vertex and has valence of $n$, and the blue lines indicate the control mesh area of $V^{ii}$, expressed by $area(H)$. The solid lines indicate the mesh at the low level of resolution, where the yellow point and the blue one are two edge points, while the dash lines indicate the mesh at the higher level of resolution.

For the lifting scheme of Loop subdivision wavelets, we need to construct a new wavelet transform with the synthesis filters $(\hat{P}^i, \hat{Q}^i)$. The control mesh at the resolution level i+1 is reconstructed as

$$V^{i+1} = \hat{\mathbf{P}}^i V^i + \hat{\mathbf{Q}}^i W^i$$

where the subdivision matrix $\hat{\mathbf{P}}^i$ of the new wavelet transform is the same as the old one, and the detail matrix $\hat{\mathbf{Q}}^i$ can be obtained from $\mathbf{Q}^i$ and $\mathbf{P}^i$ with a lifting operation as below:

$$\hat{P}^i = P^i,$$
$$\hat{Q}^i = Q^i + P^i S^i,$$

where $S^i$ is the lifting matrix.

A higher level of the control vertices is computed by the lifting scheme as follows:

$$V^{i+1} = P^i V^i + (Q^i + P^i S^i) W^i$$
$$= P^i (V^i + S^i W^i) + Q^i W^i .$$

The above equation can be rewritten as

$$\hat{V}^i \leftarrow V^i + S^i W^i ,$$
$$V^{i+1} \leftarrow P^i \hat{V}^i + Q^i W^i .$$

The vertices in $V^H$ have the same topological relationship as $V^L$. Let

$\mathbf{V}^H = (\mathbf{V}_0, \mathbf{V}_1, ?\quad \mathbf{V}_k)^T$

indicate the control vertex vector of $V^H$. According to the lifting scheme, $\mathbf{V}^H$ can be represented as

$$\mathbf{V}^H = \hat{\mathbf{P}}^A \mathbf{V}^A + \hat{\mathbf{Q}}^A \mathbf{W}^A . \tag{3.3}$$

Since $\mathbf{V}^L$ is obtained by directly subdividing $\mathbf{V}^A$, we have

$$\mathbf{V}^L = \mathbf{P}^A \mathbf{V}^A .$$

Let $f^H(v,w)$ indicate the limit surface of the triangular patch defined by $\mathbf{V}^H$. Thus The $d^{th}$ derivative at the parametric position $(v,w)$ at the higher resolution level, $f^{H^{(d)}}(v,w)$, can be obtained by multiplying the scaling functions with the control vertex vector $\mathbf{V}^H$ as below:

$$f^{H^{(d)}} = \Phi^{H^{(d)}}(v,w)\mathbf{V}^H .$$

The $d^{th}$ derivative of the surface defined by the control mesh at one lower resolution level at the same place, $f^{L^{(d)}}(v,w)$, can also be obtained Similarly.

To keep the geometric constraints, the following equation should be satisfied:

$$f^{L^{(d)}}(v,w) = f^{H^{(d)}}(v,w), \tag{3.4}$$

where $d \geq 0$ is an integer and the superscript $(d)$ indicates the $d$th order derivative of the surface $f^L(v,w)$ or $f^H(v,w)$. If $d=0$, Equation (3.4) is a position constraint, or a $d$th order derivative constraint.

We know that:

$$f^{H^{(d)}}(v,w) = \mathbf{\Phi}^{H^{(d)}}(v,w)\mathbf{V}^H ,$$

where $\mathbf{\Phi}^H(v,w) = \left\{\phi_0^H(v,w), \phi_1^H(v,w), \cdots, \phi_{k-1}^H(v,w)\right\}$ is a vecter of the scaling basis functions of the wavelet transform, composed of the box-spline basis functions for Loop subdivision. Multiplying both sides of Equation (3.3)

by $\Phi^{H^{(d)}}(v,w)$ yields

$$f^{H^{(d)}}(v,w) = f^{L^{(d)}}(v,w) + \widehat{\Psi}^{H^{(d)}}(v,w)W^A \tag{3.5}$$

where

$$\widehat{\Psi}^{H^{(d)}}(v,w) = \Phi^{H^{(d)}}(v,w)P^A S^A + \Phi^{H^{(d)}}(v,w)Q^A$$

is a vector of the wavelet basis functions.

Substituting Equation (3.5) into Equation (3.4), we obtain a necessary property of constrained wavelet basis functions of Loop subdivision

$$\widehat{\Psi}^{H^{(d)}}(v,w) = 0,$$

that is,

$$\Phi^{H^{(d)}}(v,w)P^A S^A = -\Phi^{H^{(d)}}(v,w)Q^A \tag{3.6}$$

where $S^A$ is our lifting matrix with lifting parameters for the constrained *area*$(H)$.

### 3.2.3.1 Position and derivative constraints

In Equation (3.6), **Q** is an $M\times(n+9)$ matrix. where $M = n+12$ and $n$ is the valence of the control vertex located at the local origin of *area*$(H)$, called L0 vertex, and **P** is an $M$譎 matrix obtained from Loop subdivision rules. Filters **P** and **Q** are given in Appendix C.

Since Loop' s subdivision scheme on triangular meshes is based on Box splines, Stam [33] presented the exact evaluation method of Loop subdivision surfaces based on Box splines. By means of Stam's method the scaling function vectors

$$\Phi^{i^{(d)}}(v,w) = \left\{\phi_0^{i^{(d)}}(v,w), \phi_1^{i^{(d)}}(v,w), \cdots, \phi_{k-1}^{i^{(d)}}(v,w)\right\}, \quad i = \{L, H\},$$

can be defined with the eigenbasis functions for Loop subdivision[33].

Geometric constraints that we focus on are position constraints and derivative constraints. $\Phi^{i^{(d)}}$ is used for position constraints when $d$=0, and for 1st order derivative (even normal) constraints when $d$=1.

$$\Phi^{i(d)} = M\ b^{(d)}(v,w) \tag{3.7}$$

$b^{(d)}(v,w)$ is a 12×1 column vector obtained by using a conversion from box splines to triangular Bezier patches developed by Lai[20]. M is a matching matrix.

The position and derivative constraints can be parametrized exactly by using the projection of $\mathbf{V}^{\mathrm{II}}$ onto the eigenspace of the subdivision matxix.

Each triangular patch on a regular part of the mesh is defined by twelve control vertices, as shown in Fig. 6 (Constraints are in the shaded triangle. The sequence of vertices is corresponding to $b(v,w)$).

Let $\mathbf{B} = \Phi^i P^A$ and $\Phi^H$ be the abbreviation of $\Phi^H(u,v)$ for short, and then the left-hand side of Equation (3.6) is written as

$$\Phi^i P^A S^A = BS^A = [B\mathbf{s}_0, B\mathbf{s}_1, BP^A\mathbf{s}_2 ..., BP^A\mathbf{s}_{n+8}]$$

where $\mathbf{S}_i$ is the $i^{th}$ column of the lifting matrix $\mathbf{S}^A$, and the right-hand side of Equation (3.6) as

$$-\Phi^{H^{(d)}}(v,w)Q^A \overset{\Delta}{=} C = [c_0, c_1, \cdots, c_{n+6}, c_{n+7}, c_{n+8}],$$

and then we have

$$B\mathbf{s}_i = c_i \rightarrow \sum_{j=0}^{n+5} B_j s_{ji} = c_i, \quad \text{i=0, 1, } \cdots, n+8 \tag{3.8}$$

where $s_{j,i}$ is the element in the $j^{th}$ row and the $i^{th}$ column of $S^A$.

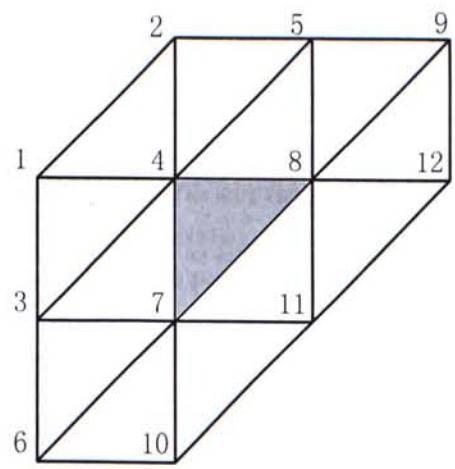



Figure 6: A triangular patch in a regular part of the mesh with constraints

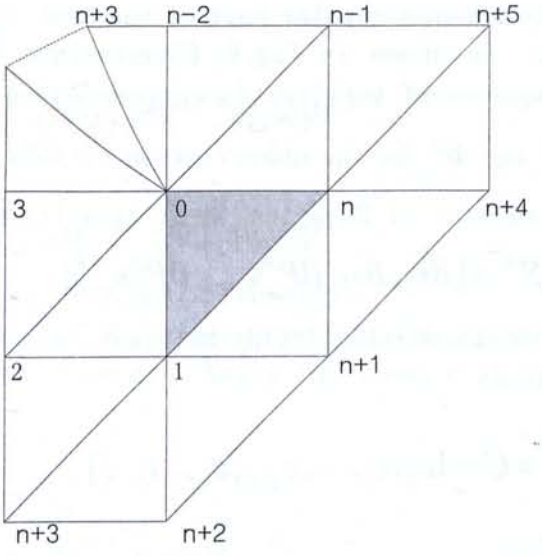


Figure 7: A triangular patch containing an extraordinary vertex with constraints

### 3.2.3.2 Geometrically-constrained Loop subdivision wavelet construction based on lifting scheme

Observing Equation (3.8) that is constructed from the geometric-constraint relationship and contains the lifting parameters inside, there may exist under-constrained situations (more than one solution), or the coefficient matrix of the linear system of equations is not of full rank. To make the solution unique, we add certain lifting operations with some local orthogonal conditions to increase the stability of the wavelet transform.

Unlike Bertram's method[1] described in Section 3.2.1, we construct a wavelet basis function $\hat{\psi}_\ell$ located at each edge point of the mesh (e.g., the yellow point in Figure 5) defining the constrained patch by orthogonalizing the corresponding lazy wavelet $\psi_\ell$ with all the $n$+6 scaling functions located at the control vertices $\mathbf{v}_i$ ($i$=0, . . . , $n$+5) of the constrained patch. Therefore, we need to obtain weights $s_{k,\ell}$, such that

$$\hat{\psi}_\ell = \psi_\ell + \sum_{k=0}^{n+5} \phi_k s_{k,\ell} .$$

where $\psi_\ell$ corresponds to edge $\mathbf{e}_l$, $\phi_k$ corresponds to vertex $\mathbf{v}_k$, $\mathbf{e}_l$ and $\mathbf{v}_k$ $\in$area(A).

Let

$$g_{i\ell} = <\hat{\psi}_\ell, \phi_i> = <\psi_\ell, \phi_i> + \sum_{k=0}^{n+5} <\phi_k, \phi_i> s_{k,\ell},$$

we minimize the square sum of all inner products $g_{i\ell}$ in area(A), such that the wavelet basis functions as orthogonal as possible to the corresponding scaling basis functions. Thus, for position constraints, we can construct a minimization problem with constraints given in Equation (3.8) as below:

$$\min \sum_{\ell=0}^{n+8} \sum_{j=0}^{n+5} (g_{j\ell})^2$$

subject to

$$\sum_{\ell=0}^{n+8} \left( \sum_{r=0}^{n+5} B_r s_{r\ell} - c_\ell \right).$$

This constrained optimization can be converted to an unconstrained optimization as

$$\min F = \sum_{\ell=0}^{n+8} \left[ \sum_{j=0}^{n+5} (g_{j\ell})^2 + \lambda_\ell \left( \sum_{r=0}^{n+5} B_r s_{r\ell} - c_\ell \right) \right] \qquad (3.22)$$

where $\lambda_\ell$ are the Lagrangen multipliers.

We can expend F with $g_{j\ell}$:

$$F = \sum_{\ell=0}^{n+8} \left( \sum_{j=0}^{n+5} (g_{j\ell})^2 + \lambda_\ell \left( \sum_{r=0}^{n+5} B_r s_{r\ell} - c_\ell \right) \right)$$

$$= \sum_{\ell=0}^{n+8} \left( \sum_{j=0}^{n+5} \left[ (g_{j\ell})^2 + \lambda_\ell B_j s_{j\ell} + \right] - \lambda_\ell c_\ell \right)$$

$$= \sum_{\ell=0}^{n+8} \left( \sum_{j=0}^{n+5} \left[ \left( <\phi_j, \psi_\ell> + \sum_{k=0}^{n+5} <\phi_k, \phi_j> s_{k,\ell} \right)^2 + \lambda_\ell B_j s_{j\ell} \right] - \lambda_\ell c_\ell \right)$$

For each unknown variable $s_{i\ell}$ or $\lambda_\ell$ in Equation (3.22), let the partial derivative of F with respect to $s_{i\ell}$ or $\lambda_\ell$ equal zero, i.e.,

$$\frac{\partial F}{\partial s_{i\ell}} = 2 \sum_{j=0}^{n+5} <\phi_i, \phi_j> \left( <\phi_j, \psi_\ell> + \sum_{k=0}^{n+5} s_{k,\ell} <\phi_k, \phi_j> \right) + \lambda_\ell B_i = 0$$

and

$$\frac{\partial F}{\partial \lambda_\ell} = \sum_{r=0}^{n+5} B_r s_{r\ell} - c_\ell = 0$$

where $\ell = 0, 1, \cdots, n+8$; $j = 0, 1, \cdots, n+5$, then the lifting filter matrix S can be computed through the above linear system of equations.

Similarly, other geometric constraints, such as first- and second-order derivative constraints, can also be handled. The procedure to compute S for normal constraints is shown in Appendix D.

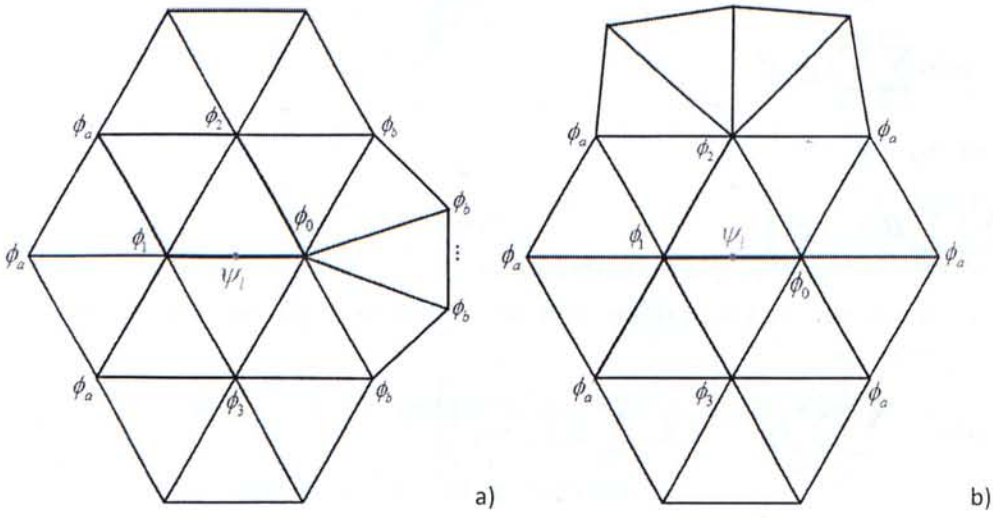



Figure 8: constructing a wavelet transform with wavelet basis function $\psi_l$ and scaling basis functions. An extraordinary vertex at different positions: a) sideward: $\phi_i$ (i=0, ···, 3, a, a, a, b, ···, b); and b) wingside: $\phi_i$ (i=0, ···, 3, a, a, a, a, a, a).

According to different positions of the extraordinary vertex in the triangular patch, as shown in Figure 8, the inner products of the wavelet basis functions and the scaling basis functions are different as below: For Figure 8 a), we have

$$\langle \psi_l, \phi_0 \rangle = a_0\delta_0 + \gamma_1\delta_1 + \frac{3}{8},$$

$$\langle \psi_l, \phi_1 \rangle = a_1\delta_1 + \gamma_0\delta_0 + \frac{3}{8},$$

$$\langle \psi_l, \phi_2 \rangle = \langle \psi_l, \phi_3 \rangle = \gamma_0\delta_0 + \gamma_1\delta_1 + \frac{1}{8},$$

$$\langle \psi_l, \phi_a \rangle = \gamma_1\delta_1,$$

$$\langle \psi_l, \phi_b \rangle = \gamma_0\delta_0.$$

For Figure 8 b), we have

$$\langle \psi_l, \phi_0 \rangle = \langle \psi_l, \phi_1 \rangle = a_0\delta_0 + \gamma_1\delta_1 + \frac{3}{8} = a_1\delta_1 + \gamma_0\delta_0 + \frac{3}{8},$$

$$\langle \psi_l, \phi_2 \rangle = \langle \psi_l, \phi_3 \rangle = \gamma_0\delta_0 + \gamma_1\delta_1 + \frac{1}{8},$$

$$\langle \psi_l, \phi_a \rangle = \gamma_0\delta_0 = \gamma_1\delta_1.$$

The discrete inner product of two scaling functions $\phi_0$ and other functions $\phi_i$, including the inner product of $\phi_0$ itself (shown in Figure 9), has four situations as follows:

$$\langle \phi_0, \phi_0 \rangle = \alpha_0^2 + \frac{n}{256} + \frac{5n}{32},$$

$$\langle \phi_0, \phi_a \rangle = \alpha_0 \gamma_0 + \frac{3}{8},$$

$$\langle \phi_0, \phi_b \rangle = \frac{1}{256},$$

$$\langle \phi_0, \phi_c \rangle = \frac{3}{128}.$$

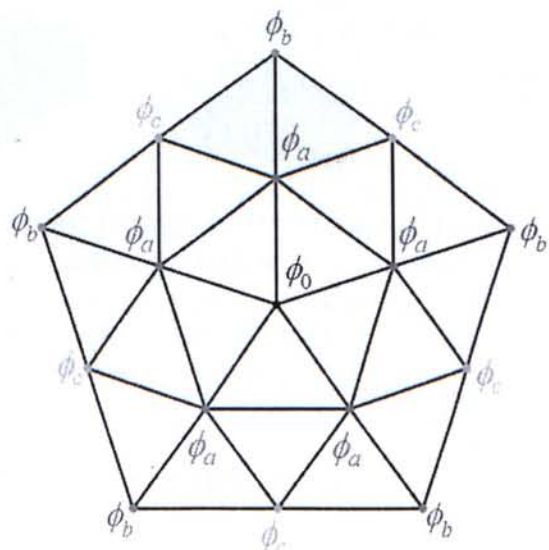



Figure 9: Discrete inner products of scaling functions of Loop subdivision wavelet transform.

Based on the computation of the lifting matrix **S** with geometric constraints, the wavelet basis functions are constructed by a prior lifting operation to the subdivision described in Equation (3.2). The wavelet synthesis procedure is defined as

$$\begin{cases} \boldsymbol{v}_i \leftarrow \boldsymbol{v}_i + \boldsymbol{s}_{ij}\boldsymbol{e}_j & \forall \boldsymbol{e}_j \in area(\mathrm{A}), \forall i = 0,...,n-1; \\ otherwise\ \boldsymbol{v}_i \leftarrow \boldsymbol{v}_i + \omega_i \mathbf{e} & \forall \boldsymbol{e}, \forall i = 0,...,3; \end{cases}$$

$$\boldsymbol{e} \leftarrow \boldsymbol{e} + \frac{3}{8}(\boldsymbol{v}_0 + \boldsymbol{v}_1) + \frac{1}{8}(\boldsymbol{v}_2 + \boldsymbol{v}_3) \quad \forall \boldsymbol{e}; \tag{3.9}$$

$$\boldsymbol{v} \leftarrow \beta(n)\boldsymbol{v} \quad \forall \boldsymbol{v};$$

$$\boldsymbol{v}_i \leftarrow \boldsymbol{v}_i + \delta_i \boldsymbol{e} \quad \forall \boldsymbol{e}, \forall i = 0,1.$$

The wavelet analysis procedure can be regarded as the inverse of Equation (3.9) fully as follows:

$$
\begin{aligned}
&v_i \leftarrow v_i - \delta_i e \qquad \forall e, \forall i = 0,1; \\
&v \leftarrow \frac{1}{\beta(n)} v \qquad \forall v; \\
&e \leftarrow e - \frac{3}{8}(v_0 + v_1) - \frac{1}{8}(v_2 + v_3) \qquad \forall e; \\
&\begin{cases} v_i \leftarrow v_i - s_{ij} e_j \qquad \forall e_j \in area(\mathrm{A}), \forall i = 0,\ldots,n-1; \\ v_i \leftarrow v_i - \omega_i e \qquad \forall e, \forall i = 0,\ldots,3. \end{cases}
\end{aligned}
\tag{3.10}
$$

Figure 10 shows the visualization of the wavelet basis functions. The position constraint is applied to the gray area of Figure 5 (right), two edge points are denoted by one yellow and one blue points in Figure 5 (right).

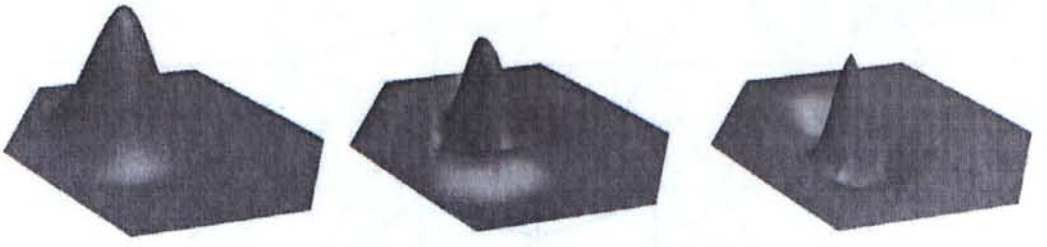


Figure 10: Basis functions of the geometry-constrained Loop subdivision wavelet transform with a position constraint. From left to right, the scaling function, edge-point wavelet function of yellow point, and edge-point wavelet function of blue point.

## 3.3 Biorthogonal subdivision wavelets with geometric constraints for quadrilateral meshes

3D models defined with quadrilateral meshes are more widely used. In this section, two typical subdivision wavelets with geometric constraints for quadrilateral meshes are discussed.

### 3.3.1 Catmull-Clark subdivision and Doo-Sabin subdivision surfaces

Catmull-Clark and Doo-Sabin subdivisions both are typical subdivision surfaces for quadrilateral meshes, but the difference between them is that Catmull-Clark subdivision is the primal subdivision, while Doo-Sabin subdivision is the dual one.

For the sake of completeness, Catmull-Clark and Doo-Sabin subdivision schemes will be introduced first in this section.

### 3.3.1.1 Catmull-Clark subdivision

Catmull-Clark subdivision is a very popular quadrilateral subdivision scheme for meshes of arbitrary topology. As a widely-used multiresolution subdivision scheme, it has attracted many researchers' attention since 1978 [6]. Catmull-Clark subdivision is defined by generalizing bi-cubic uniform B-spline [6]. The basic subdivided rule is to introduce a new vertex f' in the center for every quadrangle and a new vertex e' for every edge of the mesh, and then relocate every old vertex to v'. The subdivision mask for computing f', e' and v' is depicted in Figure 11. Base on these indices, we can write the subdivision mask of Catmull-Clark subdivision as:

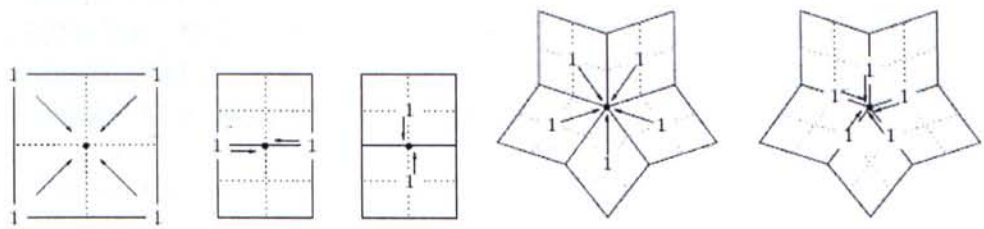



Figure 11: Local Subdivision Operators: v▷f means "take the average of all the vertices around an face point"; v▷e means "take the average of the two vertices adjacent to an adge point"; f▷e means "take the average of the two face points adjacent to an edge point"; f▷v means "take the average of all the face points around a vertex"; e▷v means "take the average of all the edge points adjacent to a vertex".

$$\boldsymbol{e} = \boldsymbol{v} \triangleright \boldsymbol{e}, \quad \forall \text{edge of the model}$$

$$\boldsymbol{f} = \boldsymbol{v} \triangleright \boldsymbol{f}, \quad \forall \text{face of the model}$$

$$\boldsymbol{v} = \alpha(n)\boldsymbol{v} + n\beta(n)\boldsymbol{e} \triangleright \boldsymbol{v} + n\gamma(n)\boldsymbol{f} \triangleright \boldsymbol{v}, \quad \forall \text{vertex}$$

$$\boldsymbol{e} = \frac{1}{2}\boldsymbol{e} + \frac{1}{2}\boldsymbol{f} \triangleright \boldsymbol{e}, \quad \forall \text{edge of the model}$$

where $n$ is the valence of the vertex $\boldsymbol{v}$, $\alpha(n) = 1 - \frac{3}{n}$, $\beta(n) = \frac{2}{n^2}$, and $\gamma(n) = \frac{1}{n^2}$.

### 3.3.1.2 Doo-Sabin subdivision

Doo-Sabin subdivision is a dual quadrilateral subdivision scheme, which is an extension of the bi-quadratic uniform B-spline surfaces to surfaces of arbitrary topology. Through a limiting process of subdivision, it produces uniform bi-quadratic B-spline surface for regular rectangular control meshes and global $C^1$-continuous surfaces for arbitrary control meshes. Since the character of dual subdivision as we discussed in previous sections that new vertices are inserted and old vertices disappeared after one subdivision step, the vertex split is the most notable characteristic of the topological refinement rules of Doo-Sabin subdivision. In a subdivision step, each vertex of valence $n$ is split into $n$ new vertices in $n$ incident faces, and the refined mesh is built with three kinds of new faces: F-faces, E-faces and V-faces, corresponding to the old faces, edges and vertices. The valences of each E-face and each vertex after one subdivision are equal to four, while all F-faces and V-faces have the same valences as the corresponding old faces and vertices.

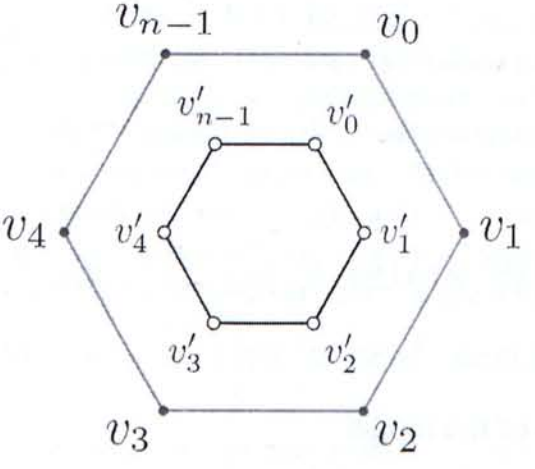



Figure 12: Mask for Doo-Sabin subdivision.

Figure 12 illustrates the mask of Doo-Sabin subdivision. For an $n$-sided face, with the notions shown in Figure 12, the geometric rules for computing new inserted vertices can be written as:

$$v_i' = \alpha v_i + \sum_{j=1}^{n-1} \beta_j v_{(i+j)\%n}$$

The linear combination coefficients of the above equation depend only on the valence of the face where the new vertices are produced. Doo and Sabin [15] gave a group of practicable coefficients as

$$\alpha = \frac{n+5}{4n},$$
$$\beta_j = \frac{3+2\cos(2\pi j/n)}{4n}.$$

### 3.3.2 Biorthogonal subdivision wavelets with geometric constraints for quadrilateral meshes

Subdivision wavelets can treat sharp features or boundary features of geometric modeling only by changing the subdivision rules for preserving sharp features [32,40]. Our constructions of geometric subdivision wavelets are different from this work. We concentrate on biorthogonal subdivision wavelets (both Catmull-Clark and Doo-Sabin) based on geometrically constrained lifting operations without any change of the subdivision rules.

#### 3.3.2.1 Biorthogonal Doo-Sabin subdivision wavelets with geometric constraints

I) Biorthogonal Doo-Sabin subdivision wavelets

In most biorthogonal subdivision wavelet constructions, the first step is to build the lazy wavelet transforms. Introducing some additional vectors, which correspond to the wavelet coefficients, in the subdivision rules is a common method to convert the original subdivision schemes to the reversible lazy wavelet synthesis processes. In the previous biorthognal wavelet constructions based on primal subdivision schemes (e.g. Loop and Catmull-Clark subdivisions), the scaling functions correspond to the re-located old vertices. With the added vectors corresponding to the wavelet coefficients, the primal subdivision rules can be naturally converted to the lazy-wavelet synthesis processes. But since Doo-Sabin subdivision is a typical dual subdivision, all the old vertices disappear after one subdivision step and the scaling space can't be established with the common method used in previous wavelet constructions based on the primal subdivisions.

To overcome this problem in the wavelet construction, we picked up the corresponding re-located vertices of the old control points in the refined mesh when establishing the scaling functions. For an old vertex

v, all new vertices inserted around it form a new V-face and the centroidof the new V-face, v' , is an appropriate vertex corresponding to v (see Figure 13). v' does not need to be stored in advance, but can be computed easily in the wavelet analysis process.

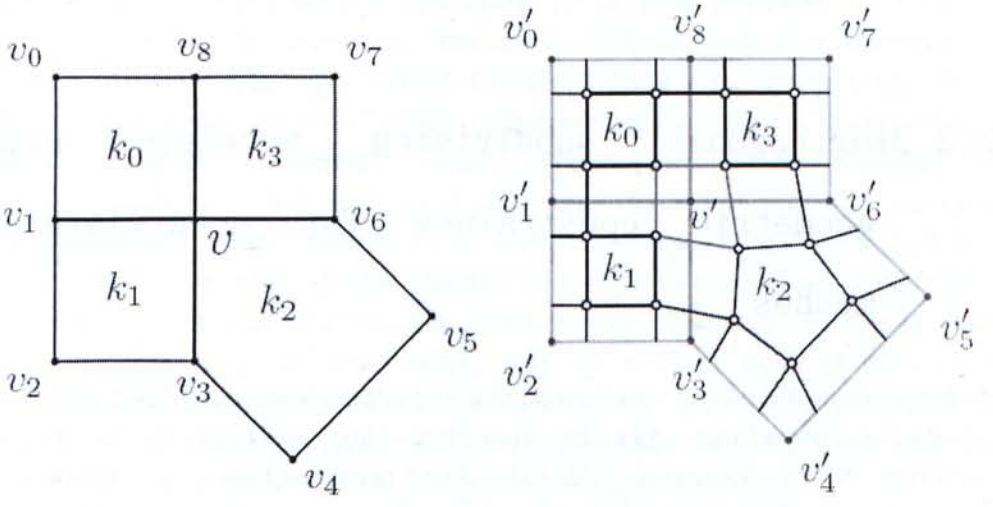



Figure 13: The corresponding vertex in the refined mesh (right) for each control point of the mesh before subdivision (left): **v** is a control point of the old mesh before subdivision, the corresponding vertex of **v** is **v'** in the refined mesh that is the centroid of the new V-face (drawn in red color).

We take $f_i'^{k_j}$ to denote the new inserted vertex which is produced by the old vertex $v_i$ in face $k_j$, and $f_i^{k_j}$ to denote the corresponding wavelet coefficient. The lazy wavelet synthesis process of Doo-Sabin subdivision can be described as follows:

$$f_i'^{k_j} = f_i^{k_j} + \alpha^{k_j} v_i + \sum_m \beta_m^{k_j} v_m$$

$$v' = \gamma v + \sum_{j=0}^{3} \left( \sum_m \eta_m^{k_j} f_m'^{k_j} \right)$$

Note that if $f_i^{k_j}$ are equal to zero initially, the above equations degenerate to the same ones as the Doo-Sabin subdivision rules.

The analysis process can be written as the inverse of the synthesis:

$$v = \left( v' - \sum_{j=0}^{3} \left( \sum_m \eta_m^{k_j} f_m'^{k_j} \right) \right) / \gamma$$

$$f_i^{k_j} = f_i'^{k_j} - \alpha^{k_j} v_i - \sum_m \beta_m^{k_j} v_m$$

To keep a good fitting quality, it is necessary to introduce the lifting scheme, which makes the wavelets orthogonal with the scaling functions. In a F-face $k$ with valence $n$, for a lazy wavelet $\psi_l$, a new wavelet $\psi$ is constructed as the linear combination of the lazy wavelet and the

$n$ scaling functions located at $\mathbf{v}_i$' ($i$=0, ···, $n$−1) which are the centroids of incident V-faces of the F-face $k$. It can be represented as

$$\psi = \lambda(\psi_l + \sum_{i=0}^{n-1} \omega_i \phi_i) \tag{3.11}$$

which satisfies that $\langle \psi, \phi_i \rangle = 0, \ (i = 0,1,...,n-1)$.

Discrete inner products are used to simplify the computation, and the discrete masks of basis functions are shown in Figure 14, where:

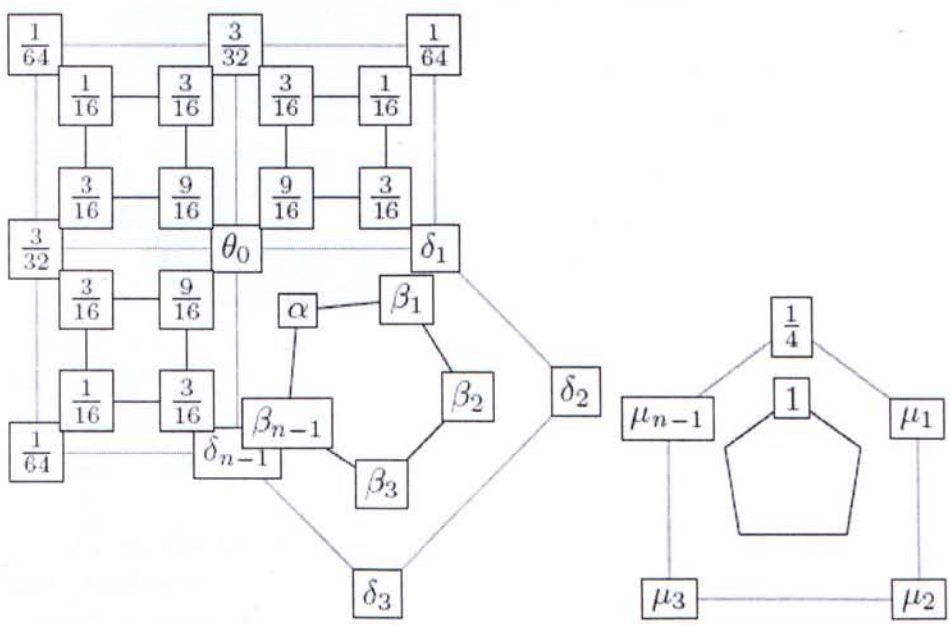
$$\alpha = \frac{n+5}{4n}, \ \beta_i = \frac{3+2\cos(2\pi i/n)}{4n}, \ i = 1,2,...,n-1,$$

$$\delta_1 = \delta_n - 1 = \frac{\beta_1}{4} + \frac{3}{64} = \frac{\beta_{n-1}}{4} + \frac{3}{64},$$

$$\delta_i = \frac{\beta_i}{4}, \ i = 2,...,n-2,$$

$$\theta_0 = \frac{\alpha}{4} + \frac{27}{64}$$

and $\mu_i$(i=1, ···, n-1) can be computed by $\alpha$ and $\beta_i$.




(a) $\phi_0$ (b) $\psi_l$

Figure 14: Discrete basis functions

With the orthogonal coefficients $\omega$, the synthesis process based on lifting scheme is obtained as follows:

$$
\begin{aligned}
&f \leftarrow \lambda f, \forall f;\\
&v_i \leftarrow v_i + \omega_i f, \quad \forall f, i = 0,...,n-1;\\
&f \leftarrow f + \alpha v + \sum_{i=1}^{n-1} \beta_i v_i, \quad \forall f;\\
&v \leftarrow \gamma v, \qquad \forall v;\\
&v_i \leftarrow v_i + \eta_i f, \quad \forall f.
\end{aligned}
\tag{3.12}
$$

The wavelet analysis can be obtained by reversing the above synthesis rules:

$$
\begin{aligned}
&v_i \leftarrow v_i - \eta_i f, \quad \forall f;\\
&v \leftarrow v/\gamma, \quad \forall v;\\
&f \leftarrow f - \alpha v - \sum_{i=1}^{n-1} \beta_i v_i, \quad \forall f;\\
&v_i \leftarrow v_i - \omega_i f, \quad \forall f, i = 0,...,n-1;\\
&f \leftarrow f/\lambda, \quad \forall f.
\end{aligned}
\tag{3.13}
$$

Figure 15 shows the visualization of the basis functions of our wavelet construction. The first row represents the scaling functions and the bottom row shows the lifted wavelets corresponding to the new vertices.

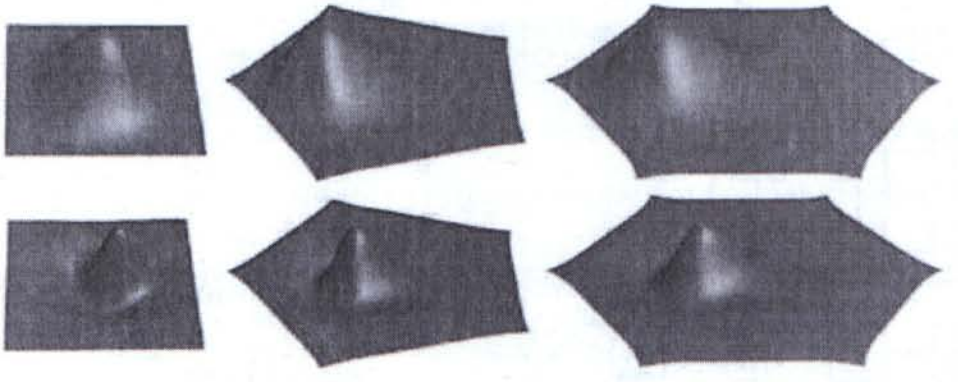


Figure 15: Visualization of the basis functions: the scaling functions (top) located in the faces with valences 4(left), 5(middle) and 6(right); the lifted wavelets (bottom) corresponding to the new vertices produced in the faces with valences 4(left), 5(middle) and 6(right).

### II) Subdivision wavelets with constraints

To construct geometric constraints, we need to introduce the relationship between control points at adjacent two resolution levels first. Take $V^H$ to denote the control point set of a geometric model at a high resolution level, and $V^A$ the result point set after the wavelet analysis process applied to $V^H$. Subdivide $V^A$ and obtain control set $V^L$ at the same low resolution level as $V^A$. According to DOO-Sabin subdivision rules and the wavelet transform described in Equations (3.12) and (3.13), vertex $v^h \in V^H$ and $v^l \in V^L$ have such relation:

$$v^h = \lambda f + v^l + \alpha L(v_0) + \sum_{j=0}^{n-1} \beta_j L(v_j), \tag{3.14}$$

where f is the wavelet coefficient associated with $v^h$ and L(v) indicates the result of the lifting operations applied to $v_j \in V^\lambda$ $(j=0, 1, \cdots, n-1)$, which is represented as

$$L(v) = \sum_{i=0}^{3} \sum_{j=0}^{n_i-1} \omega_{ij} f_{ij}$$

$f_{ij}$ means the $j^{th}$ face point wavelet in the $i^{th}$ face, $\omega_{ij}$ is the lifting coefficient of each $f_{ij}$.

The discrete geometric constraints have such relations shown as following:

$$f^{i(d)}(u,v) = f^{i+1(d)}(u,v) \tag{3.15}$$

where $f^{i(d)}(u,v)$ denote the $d^{th}$ derivative with respect to u or v at a point of the model at the $i^{th}$ resolution level. When d=0, it is for position constraint.

Shown in Figure 16, an gray area Ω of the model is controlled by vertex set $V^H$ at level i+1, where n is the valence of F-face constructed by $V^H$. Based on the approach of precise parametrization of uniform Doo-Sabin subdivision surfaces [38], the $d^{th}$ derivative at a point in Ω is computed as:

$$f^{i+1(d)}(u,v) = \Phi^{(d)}(u,v) \cdot V^H = \sum_{k=0}^{n+4} \phi_k^{(d)}(u,v) v_k^h \tag{3.16}$$

We can obtain vertex set $V^\lambda$ in lower resolution level by applying the wavelet analysis process to the model. Corresponding to $V^H$, $V^L$ is the control point set of Ω at the $i^{th}$ resolution level which can be obtained from $V^\lambda$ by the subdivision process. So the discrete geometric constraints at level i is computed as:

$$f^{i(d)}(u,v) = \Phi^{(d)}(u,v) \cdot V^L = \sum_{k=0}^{n+4} \phi_k^{(d)}(u,v) v_k^l \tag{3.17}$$

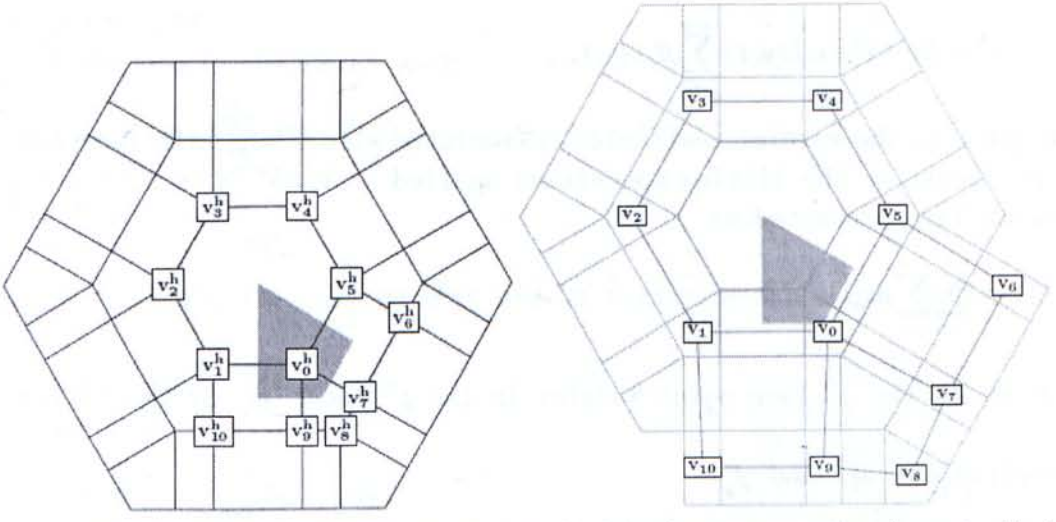



Figure 16: The geometric constraints are applied in the gray area that is controlled by the vertices of $V^H$ at resolution level i+1 (left). Each vertex of $V^A$ linked by the red lines is produced by a wavelet analysis process (right).

Taking equations (3.15) and (3.16) into Equation (3.14) with the relation of $V^H$ and $V^L$ in Eq. (3.13), we have following equation:

$$\sum_{k=0}^{n+4} \lambda_k \phi_k^{(d)}(u,v) f_k = -\sum_{k=0}^{n+4} \phi_k^{(d)}(u,v)[\alpha L(v_{k_0}) + \sum_{j=1}^{n-1} \beta_j L(v_{k_j})]$$

where $f_k$ (k=0,..,n+4) are the wavelet coefficients associated with the vertices of $V^H$, and $v_{k_j} \in V^A$.

For each wavelet coefficient $f_k$ in the lifting area of $V^A$, the following linear equation is established:

$$\sum_{k=0}^{n-1} \omega_k^f \delta_k^f = -\theta^f$$

If corresponding control point $v_k^h \in V^H$, then $\theta^f = \lambda_k \phi_k^d(u,v)$, otherwise $\theta^f = 0$.

For each constraint we can obtain a linear equation with the lifting parameters for each of the related wavelet coefficients, and all the linear equations related to the same wavelet coefficient f could form a linear system of equations to determine the lifting parameters of f corporately.

The under-constrained situation, can be treated as the idea similar to that introduced in section 3.2.3.2.

### 3.3.2.2 Biorthogonal Catmull-Clark subdivision wavelets with geometric constraints

Biorthogonal Catmull-Clark subdivision wavelets

Wang[37] constructs an efficient biorthogonal wavelet construction for the generalized Catmull-Clark subdivision by using local lifting operations. Lazy wavelet $\Psi^f$ or $\Psi^e$ should be orthogonal with local scaling functions around it (shown in Figure 17).

According to the general definition of the lifting scheme for subdivision wavelets, we introduce two lifting operations here, which are similar to the local subdivision operators shown in Figure 11, $f \trianglerighteq v$ denotes the lifting operations applied to all related face-point wavelets of one vertex, $e \trianglerighteq v$ denotes the lifting operations applied to all the related edge-point wavelets of one vertex. Let L(v) be the sum of these two operations for short, then we can construct our wavelet synthesis algorithm as:

$$v = v + L(v), \quad \forall v$$

$$e = e + v \triangleright e, \quad \forall e$$

$$f = f + v \triangleright f, \quad \forall f$$

$$v = \alpha(n)v + n\beta(n)e \triangleright v + n\gamma(n)f \triangleright v, \quad \forall v$$

$$e = \frac{1}{2}e + \frac{1}{2}f \triangleright e, \qquad \forall e$$

And the analysis process is obtained by inverting synthesis shown as followed:

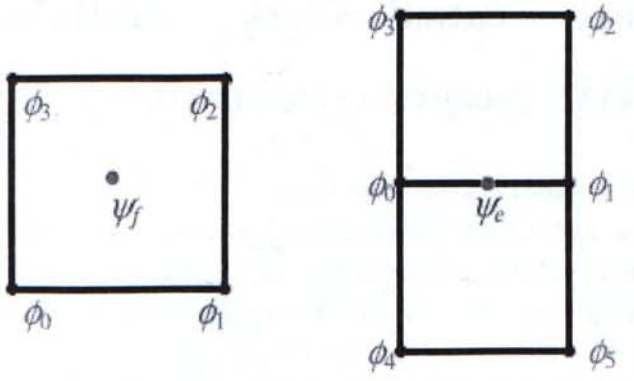
$$e = 2e - f \triangleright e, \quad \forall e$$

$$v = (v - n\beta(n)e \triangleright v - n\gamma(n)f \triangleright v)/\alpha(n), \quad \forall v$$

$$f = f - v \triangleright f, \quad \forall f$$

$$e = e - v \triangleright e, \quad \forall e$$

$$v = v - L(v), \quad \forall v$$




**a** orthogonalize $\psi_f$ **b** orthogonalize $\psi_e$

Figure 17: Using local scaling functions to orthogonalize lazy wavelets.

Catmull-Clark subdivision wavelets with geometric constraints

We construct geometric constraints by means of the relationship between the control points at adjacent resolution levels. Let $V^H$ be the control vertex set of a geometric model at a high resolution level, and $V^A$ the result vertex set after the wavelet analysis process applied to $V^H$. The control vertex set $V^L$ at the one lower resolution level is the subdivided result applied to $V^A$ of Catmull-Clark subdivision. $V^L$ and $V^H$ have a bijective relation. And according to the definition of the wavelet transforms and Catmull-Clark subdivision rules, it can be expressed as:

$$V^H = V^L + V^W + \left\{ \sum_{v^a \in area(v)} L(v^a) \mid \forall v \in V^H \right\} \qquad (3.18)$$

where $V^W$ is the related wavelet set obtained by the wavelet analysis process of $V^H$ (see Figure 18). area(v) denotes a subset of $V^A$ which will produce $v \in V^H$ with a wavelet synthesis process: for a face point $f^h \in V^H$, area($f^h$) includes four points of $V^A$ composing the face; for a edge point $e^h \in V^H$, area($e^h$) includes six points of $V^A$ which are around $e^h$; for a vertex point $v^h \in V^H$ with valence n, it has a correspondence $v^a$ according to the wavelet analysis algorithm and area($v^h$) includes 2n points of $V^A$. For these three types of vertices of $V^H$, the above relation between $V^L$ and $V^H$ has concrete formats given in Table 1, where $a(n)=1-7/4n$, $b(n)=3/(2n^2)$ and $c(n)=1/(4n^2)$.

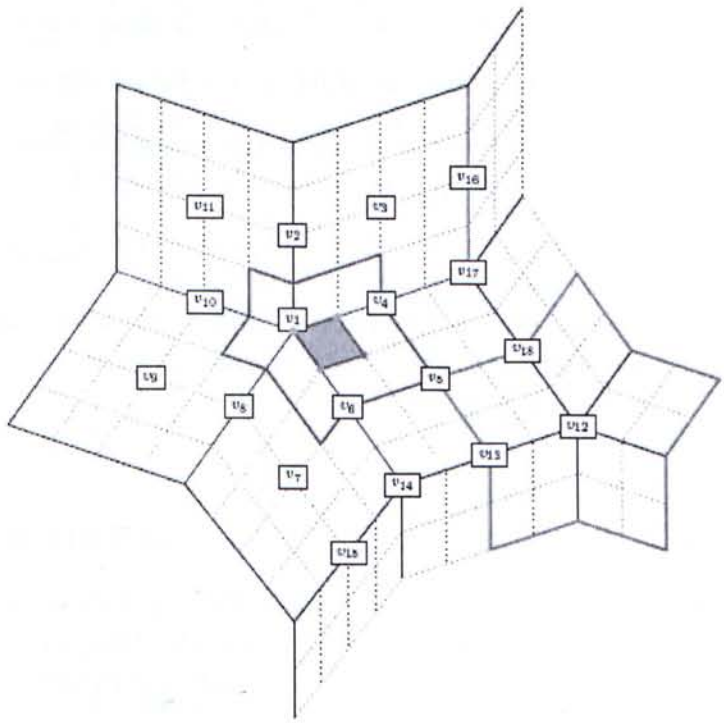



Figure 18: The control vertices of $V^H$ are encircled by the blue polyline, and the lift area of a vertex $v_{12} \in V^A$ is encircled by the brown polyline.

| Vertex Type | $V^H$ | $V^L$ | $V^W$ |
|---|---|---|---|
| Face point | $f^h$ | $f^l$ | $f$ |
| Edge Point | $e^h$ | $e^l$ | $\frac{1}{2}e + \frac{1}{4}(f_0 + f_1)$ |
| Vertex Point | $v^h$ | $v^l$ | $\beta(n)\sum_{i=1}^{n} e_i + \gamma(n)\sum_{i=1}^{n} f_i$ |
| | | | $\sum_{v^a \in area(v)} \mathbb{L}(v^a)$ |
| | | | $\frac{1}{4}\sum_{i=1}^{4} \mathbb{L}(v_i^a)$ |
| | | | $\frac{3}{8}\sum_{i=1}^{2} \mathbb{L}(\tilde{v}_i^a) + \frac{1}{16}\sum_{i=1}^{2} \mathbb{L}(\hat{v}_i^a)$ |
| | | | $a(n)\mathbb{L}(v_0^a) + b(n)\sum_{i=1}^{n} \mathbb{L}(\tilde{v}_i^a) + c(n)\sum_{i=1}^{n} \mathbb{L}(\hat{v}_i^a)$ |

Table 1: The relation between the face points, edge points and vertex points at adjacent resolution level: e and f are the wavelet coefficients associated with an edge point $e^h$ and a face point $f^h$. For a face point, $v_i^a (i = 1,2,3,4)$ are four points of the face where $f^h$ is produced; for an edge point, $\tilde{v}^a$ and $\hat{v}^a$ represent the two end points of the edge and four vertices of the two faces that share the edge; for a vertex point, $v_0^a$ is the correspondence of $v^h$ in $V^A$, and $\tilde{v}^a$ and $\hat{v}^a$ represent the 1-ring and 2-ring vertices of $v^a$.

To keep the geometric constraints, we let $f^{h^{(d)}}(u,v)$ denote the $d^{th}$ derivative at a point with parameter (u, v) at a high resolution level, and $f^{l^{(d)}}(u,v)$ the corresponding $d^{th}$ derivative at the point at the one lower resolution level. They need to satisfy that:

$$f^{h^{(d)}}(u,v) = f^{l^{(d)}}(u,v) \tag{3.19}$$

The $d^{th}$ order derivative at a point of the Catmull-Clark subdivision surface can be evaluated by means of the exact parameterization of Catmull-Clark subdivision surfaces [34]. We have:

$$f^{h^{(d)}}(u,v) = G^H \cdot \Phi^{(d)}(u,v)$$

where $\Phi(u,v)$ is the vector of eigenbasis functions defined with the bicubic B-splines, and $G^H$ is the projection of $V^H$ into the eigenspace of the Catmull-Clark subdivision matrix. Taking T to denote the invertible matrix including the eigenvectors of the corresponding subdivision matrix, we have

$$f^{h^{(d)}}(u,v) = (V^H T^{-T}) \cdot \Phi^{(d)}(u,v) = V^H \Gamma(u,v) \tag{3.20}$$

where $\Gamma(u,v) = T^{-T}\Phi^{(d)}(u,v)$.

From Equation (3.19), we obtain the equation:

$$V^L \cdot \Gamma^{(d)}(u,v) = V^H \cdot \Gamma^{(d)}(u,v) \tag{3.21}$$

Equation (3.18) gives the relation of control vertices $V^H$ and $V^L$, with Equations (3.18) and (3.21) we can construct the lifting scheme for derivative constraints.

The mask of the lifting scheme is illustrated in Figure 19, and Figure 19(c-e) show the relationship between the prescribed derivative constraints and the control points at two adjacent resolution levels. As shown in Figure 19(a)-(b), for a vertex $v_k \in V^l$ with valence n, the lifting operations can be described as:

$$f \underline{\triangleright} v = \sum_{i=0}^{n-1} \omega_i^k f_i$$

$$e \underline{\triangleright} v = \sum_{i=0}^{n-1} \eta_i^k e_i + \sum_{i=0}^{2n-1} \lambda_i^k g_i$$

For a wavelet coefficient f associated with a face point $f^h$ at the high resolution level, the linear equation of four lifting coefficients of f can be established as follows:

$$a_0\omega_0 + a_1\omega_1 + a_2\omega_2 + a_3\omega_3 = -d$$

where $\omega$ is the coefficient of f and d depends on contribution of the

wavelet coefficient f to $f^{h(d)}(u, v)$. $a_k$ equals the contribution of $v_k$ to the value of $f^{h(d)}(u, v)$ if $a_k$ belongs to $V^1$.
And the one of six lifting coefficients for an edge point wavelet e associated with the edge point $e^h$ is:

$$b_0\lambda_0 + b_1\lambda_1 + c_0\eta_0 + c_1\eta_1 + c_2\eta_2 + c_3\eta_3 = -m$$

Where $b_k$ and $c_k$ are the contributions of corresponding vertices and m is the contribution of the edge point wavelet coefficient to $f^{h(d)}(u, v)$.

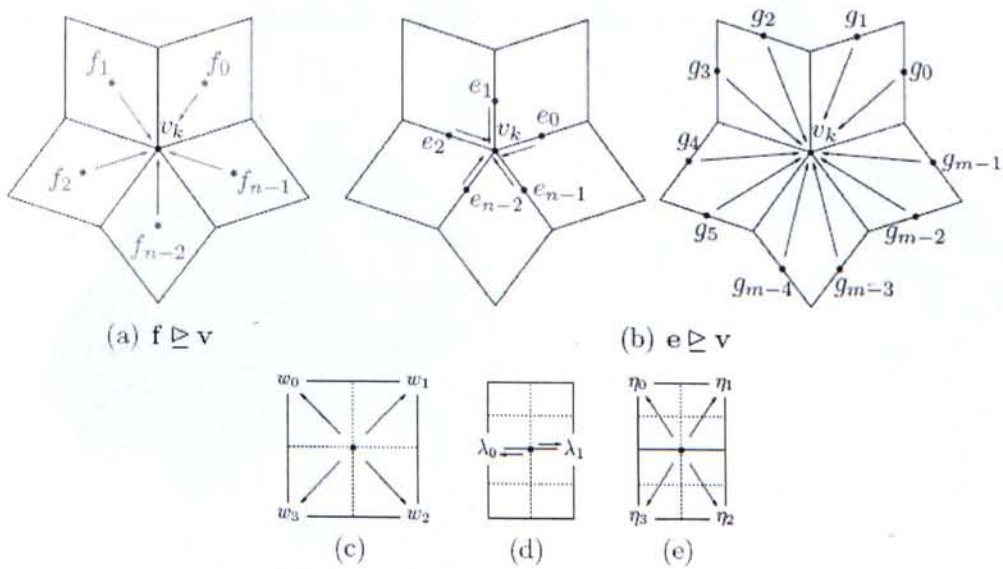



Figure 19: lifting masks.

Some visualization of the scaling and wavelet basis functions, $\phi$, of geometric-constraint based Catmull-Clark subdivision wavelets with a position constraint are shown in Figure 20. The position constraint is applied to the center of the gray area in Figure 17, and the gray area is a quadrilateral face (encircled by the green polyline) defined by a vertex point, a face point and two edge points of V H that are denoted by one yellow, one red and two green points in Figure 17. The scaling function corresponding to the vertex point) and three wavelet functions (corresponding to the face point and two edge points) are shown in Figure 20.

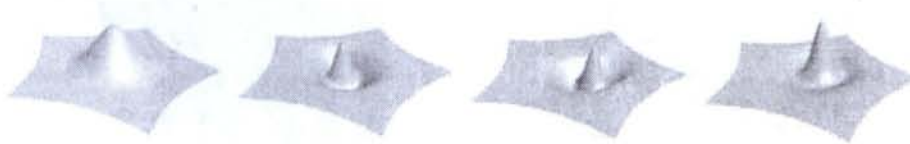


Figure 20: Basis functions of Catmull-Clark subdivision wavelet transform with a position constraint. From left to right, the scaling function, two edge-point wavelet functions and the face-point wavelet function are displayed.

The under-constrained situation, can be treated as the idea similar to that introduced in section 3.2.3.2.

# 4 Experiments and results

**Experiment 1. The multiresolution representation of the model Venus by Loop-subdivision-based wavelet transform with constraints:**

In Figure 21, the constraint point is assigned at the nose of Venus. (a) the original model at level 5; (b) the model with a constraint at the blue point on the nose, at level 5; (c) the model at level 0 without any constraints; (d) the model at level 0 with the position constraint. The two blue points indicate the positions of the point constraints in (b) and (d), respectively.

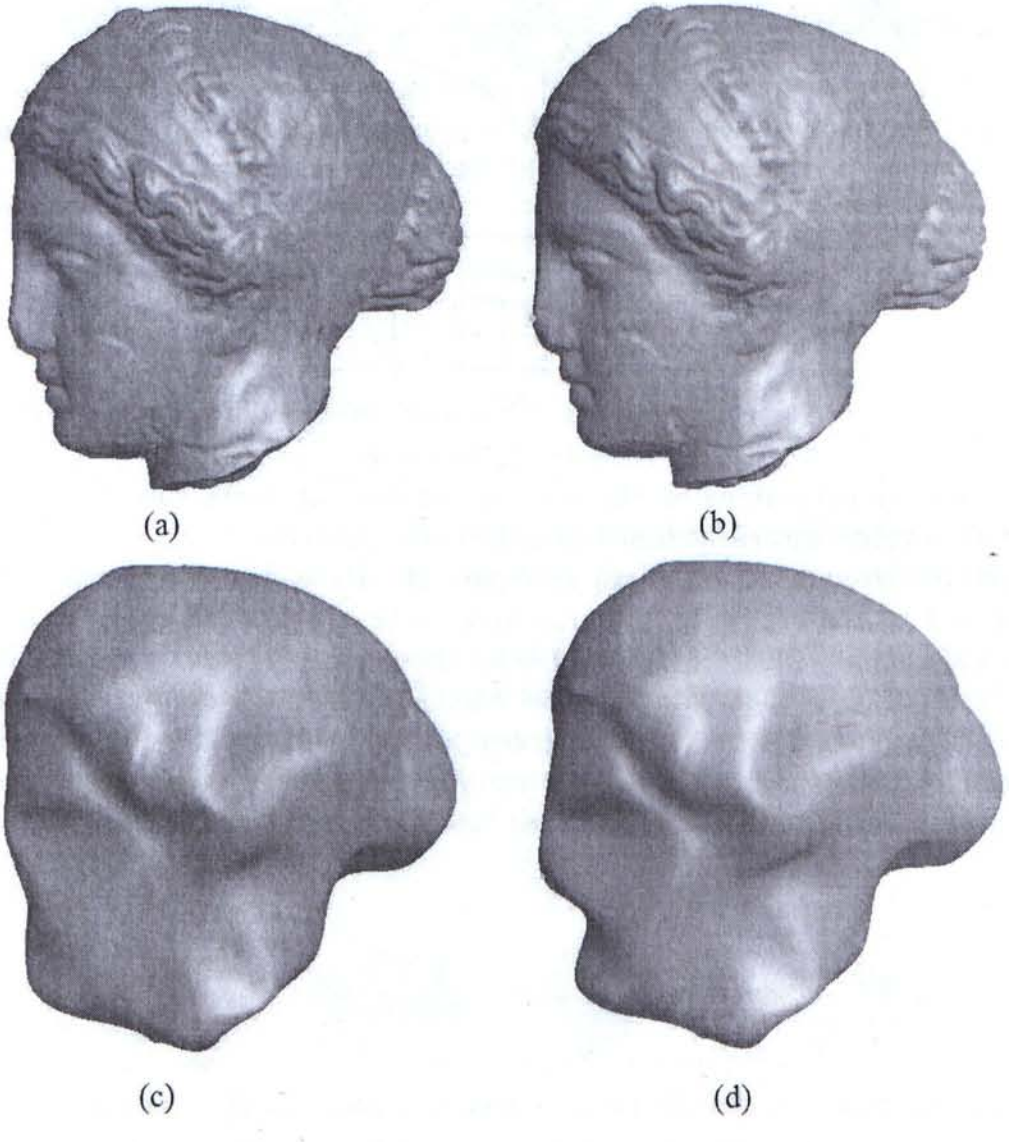



Figure 21. multiresolution representations of the model Venus with a position constraint.

**Experiment 2. The multiresolution representations of the model Feline by Loop-subdivision-based wavelet transform with constraints:**

In Figure 22, two constraint points are assigned at the ears of the feline. (a) the original model at level 5; (b) the model at level 0 without any constraints; (c) the model with constraints at level 5; (d) the model at level 0 with 2 position constraints. The four blue points indicate the positions of the point constraints in (c) and (d), respectively.

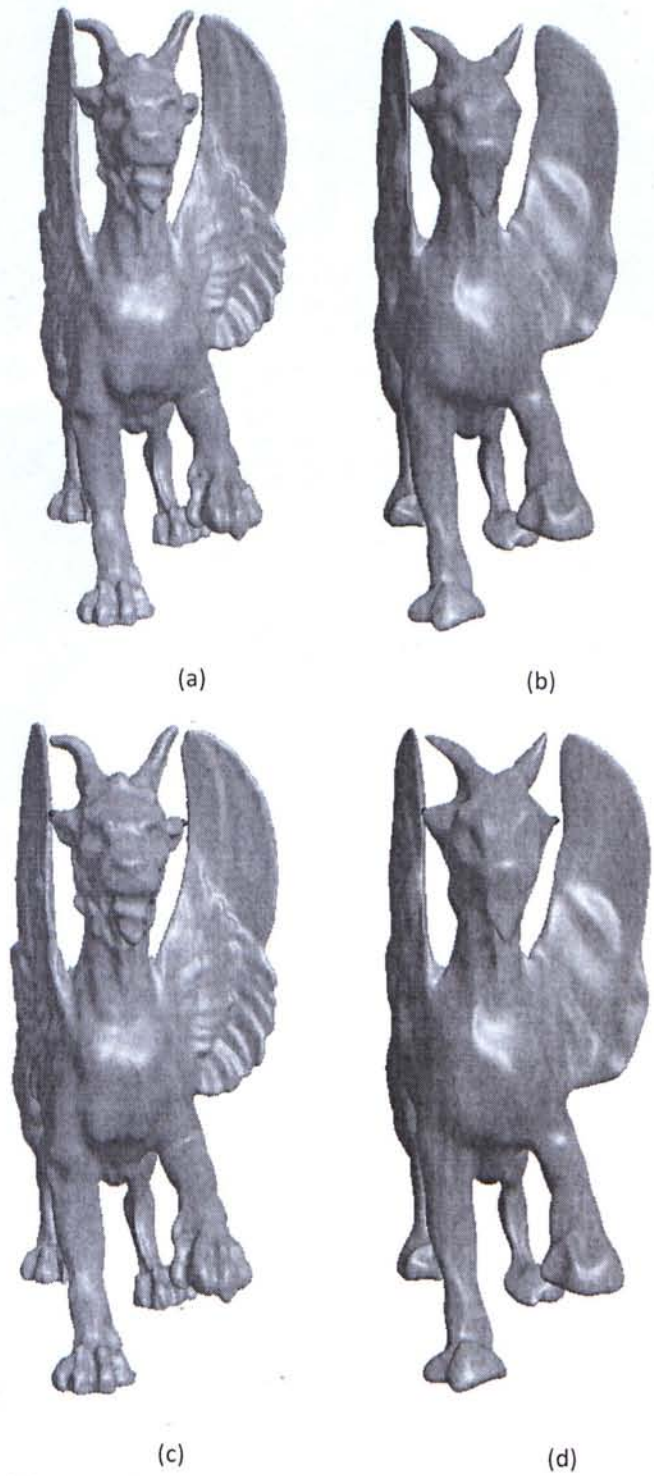



Figure 22. multiresolution representations of the model Feline with a position constraint

**Experiment 3. Doo-Sabin subdivision wavelet analyses of the model Venus from level 5 to level 0**

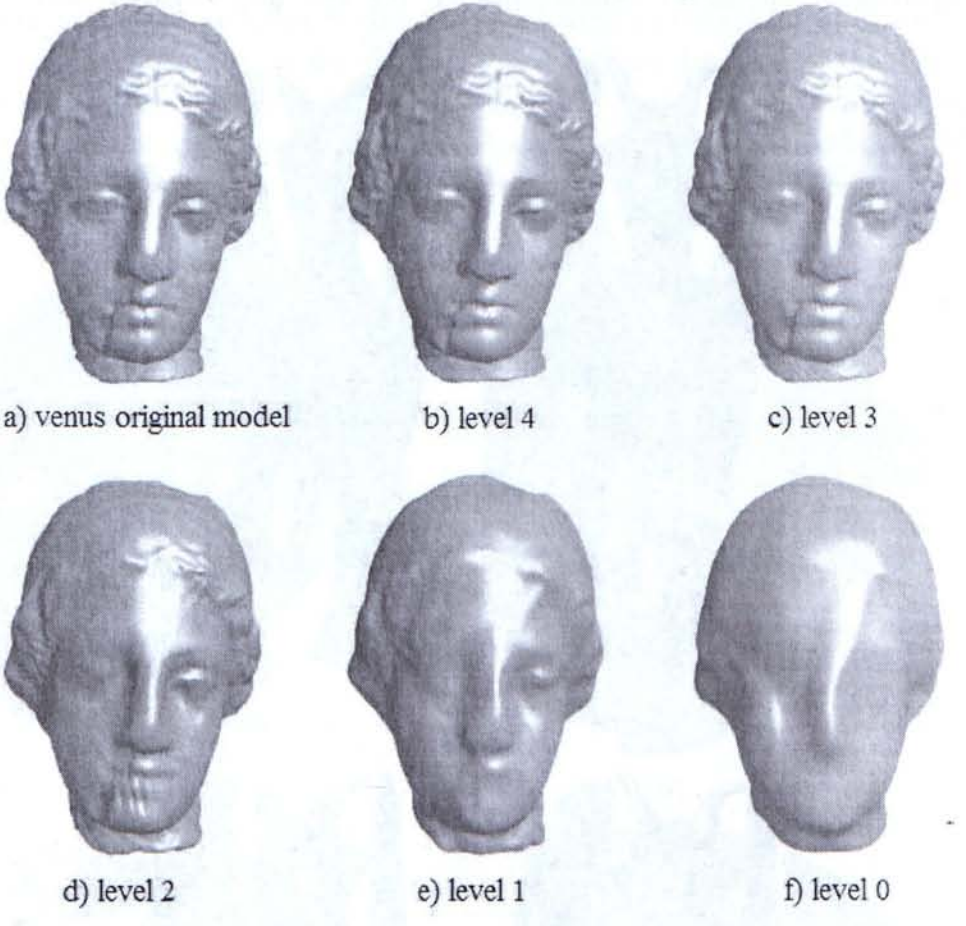



Figure 23. Doo-Sabin subdivision wavelet analyses of the model Venus from level 5 to level 0

**Experiment 4. Doo-Sabin subdivision wavelet analyses of the model Horse from level 5 to level 0**

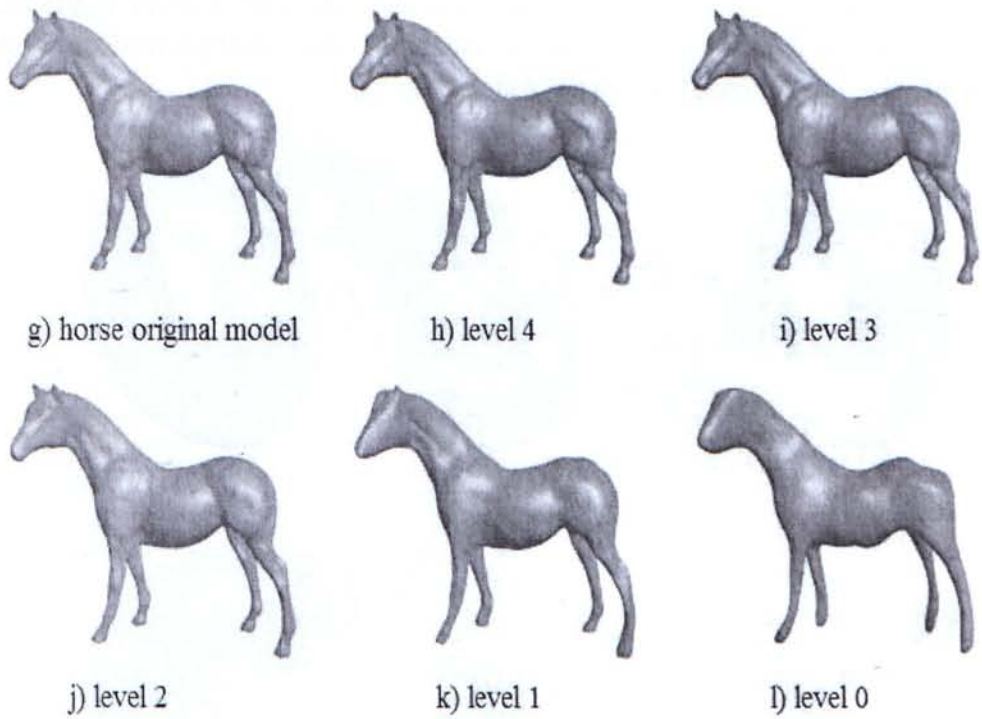



Figure 24. Doo-Sabin subdivision wavelet analyses of the model Horse from level 5 to level 0

**Experiment 5. Improved Catmull-Clark surface approximation with 2nd-order-derivative constraints:**

In Figure 25, (a) The original model (left), the lower-resolution models of the wavelet analyses with (middle) and without (right) the constraints. The red frame shows the zoom-in view of the mesh of the original model, and the five blue points indicate the 5 position constraints and 2nd-order-derivative ones; (b) The longitudinal section views of the corresponding three models in (a).

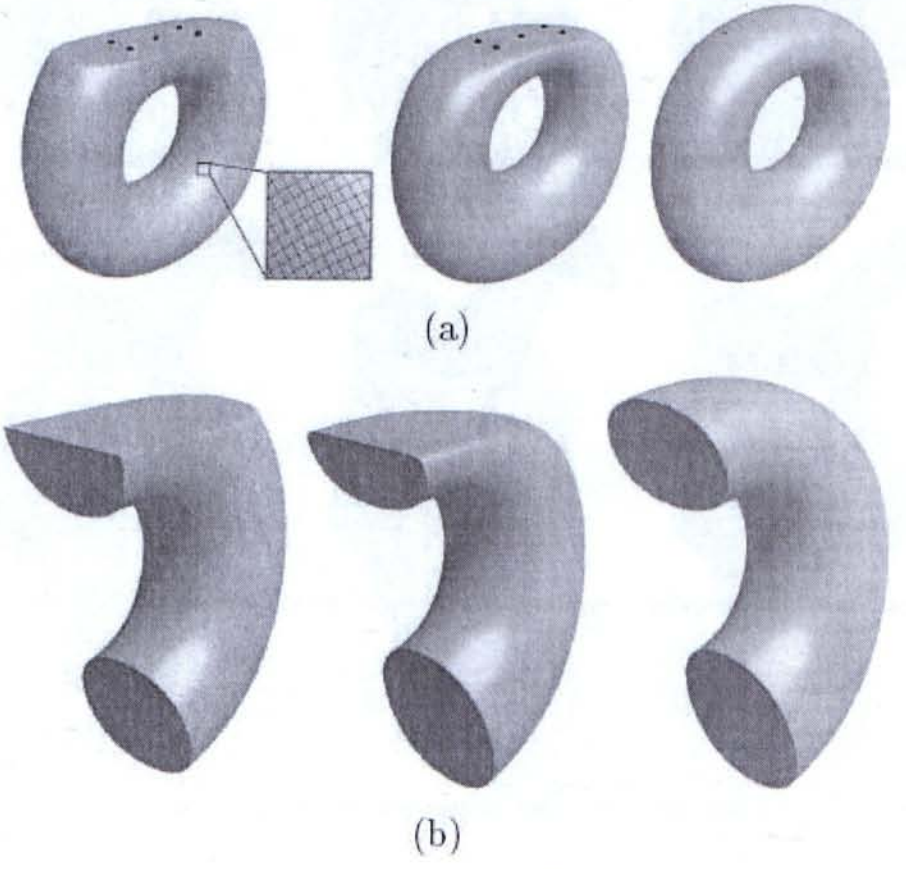
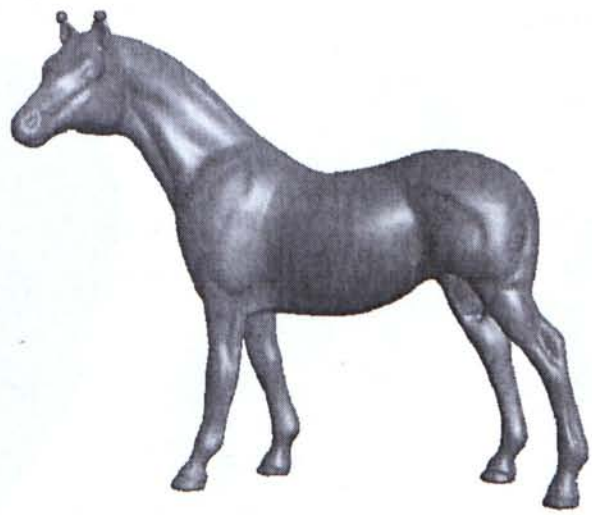



Figure 25. Improved surface approximation by Catmull-Clark subdivision-based wavelets with 2nd-order-derivative constraints

**Experiment 6. The multiresolution representations of the horse model by Catmull-Clark subdivision-based wavelet with position and derivative constraints:**

In Figure 26, (a) the original model at level 5; (b) the model at level 0 without any constraints; (c) the model at level 0 with 2 position & derivative constraints. Two red points indicate the positions of the point constraints.



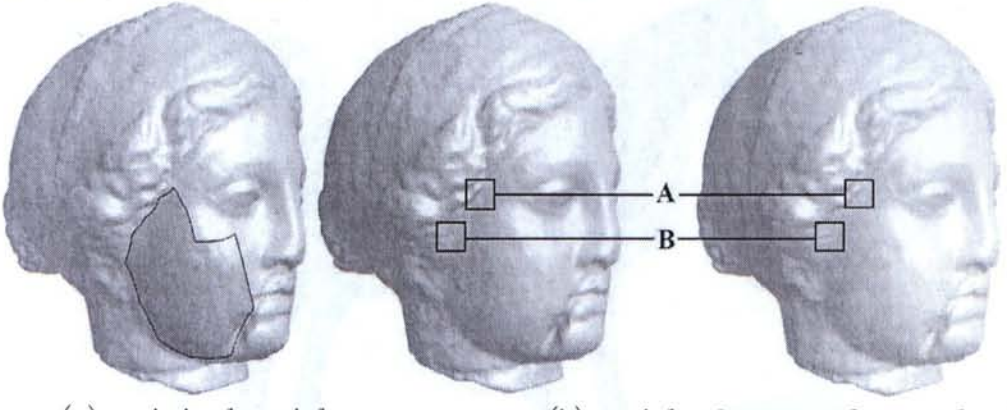
(a) horse original model

(b) (c)

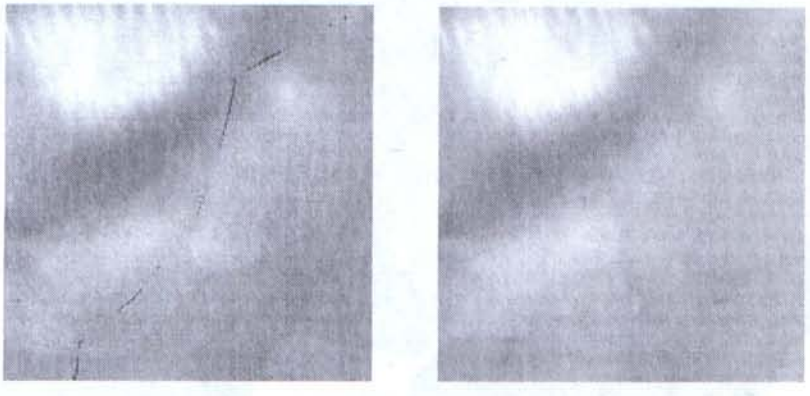
Figure 26. The multiresolution representations of the model horse with two position and derivative constraints.

**Experiment 7. Removing gaps at joints of different-resolution parts of a model:**

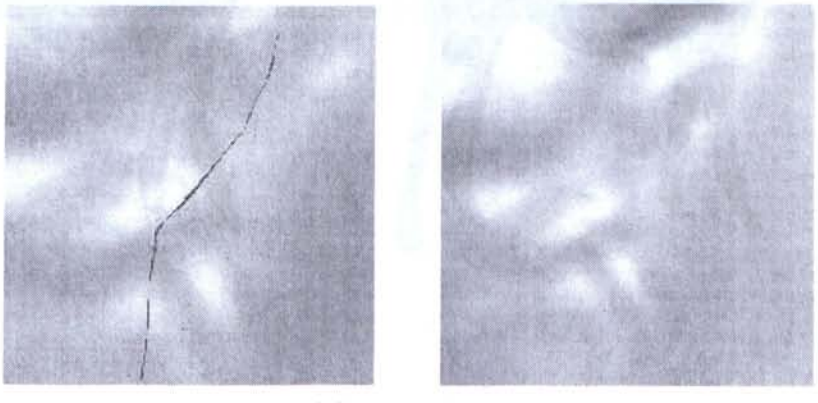
(a) The original model at the resolution level 5. According to the details, the smoother part (encircled by the blue curve used as the curve constraint) can be represented at the resolution level 3, and the curve marks the joint of the two parts at different resolution levels; (b) Wavelet analysis is applied to the smoother part of the model with (right) and without (left) the curve constraint. We give the contrast of the close-ups of the areas A and B (in (c) and (d)) of the model obtained by the common subdivision analysis (left) and GCLS-based wavelet analysis (right), respectively.




(a) original model (b) model after wavelet analysis



(c) area A



(d) area B

Figure 27. Elimination of gaps at the joints of different-resolution parts of a model by Catmull-Clark subdivision-based wavelet transform with curve constraints.

**Experiment 8. Improved Doo-Sabin surface approximation with 2nd-order-derivative constraints:**

(a) the original model at resolution level 8 (left); (b) & (c) the lower-resolution models (resolution level 3) produced by the wavelet analysis with and without the constraints, respectively; (d) the top plane of the original model; (e) the shape of the top plane after the wavelet analysis with the constraints; (f) the bottom plane of the original model; (g) the shape of the bottom plane after the wavelet analysis without the constraints;

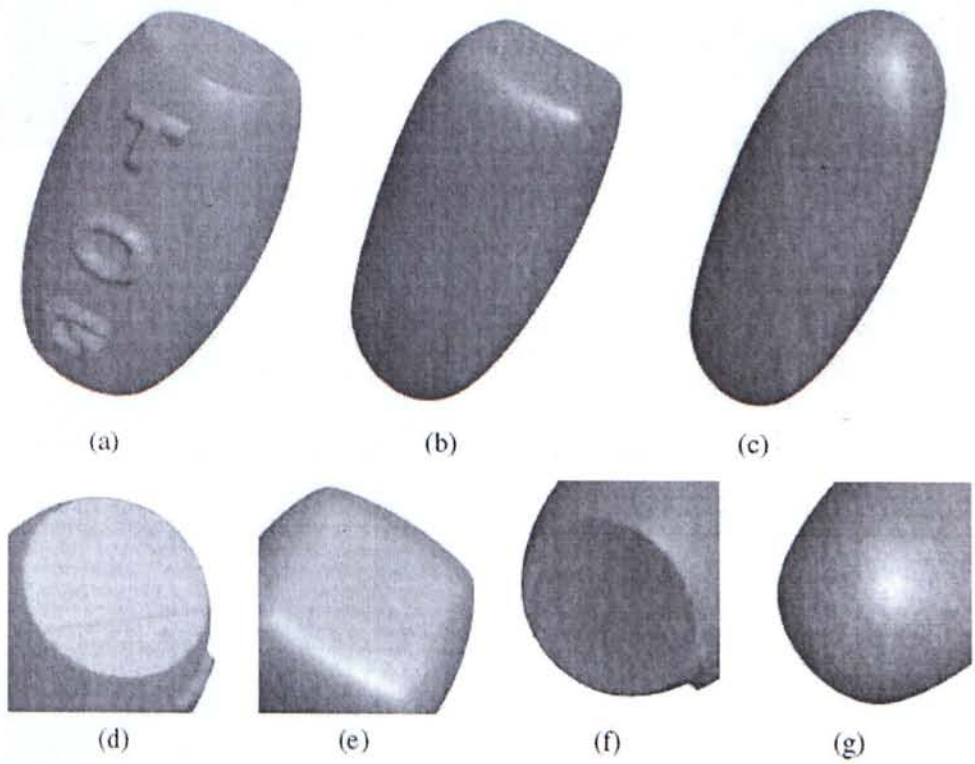


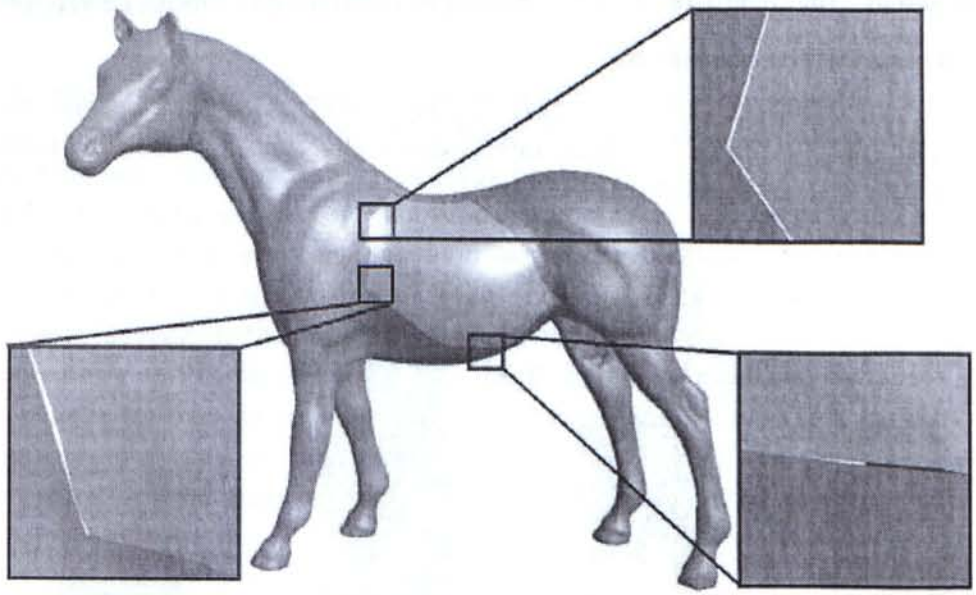

Figure 28. Improved surface approximation with 2nd-order-derivative constraints

**Experiment 9. Elimination of gaps at the joints of different resolution parts of a model by Doo-Sabin subdivision based wavelet transform with curve constraints:**

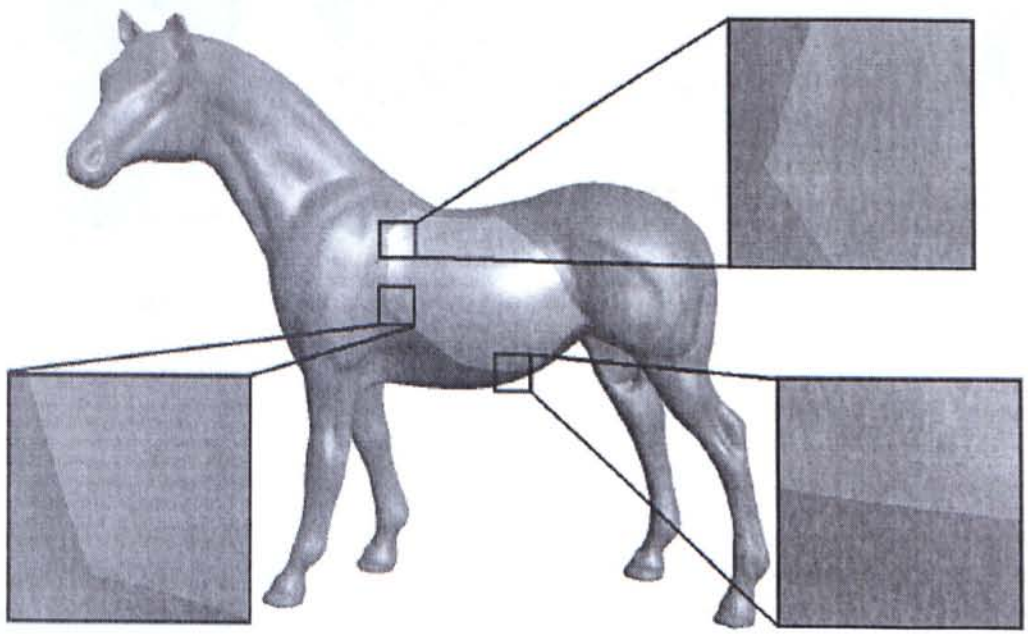
The smoother part (encircled by the green curve used as the curve constraint) can be represented at the resolution level 3, and the curve marks the joint of the two parts at different resolution levels.

This experiment is base on our method of Doo-Sabin subdivision wavelet with constraints. The wavelet analysis is applied to the smoother part of the model with (a) and without (b) the curve constraints in Figure 29. To show the results clearly, we zoom in the gaps at the joints of different-resolution parts of the model.

The original model "horse" at the 5th resolution level.



(a) wavelet analysis without constraints

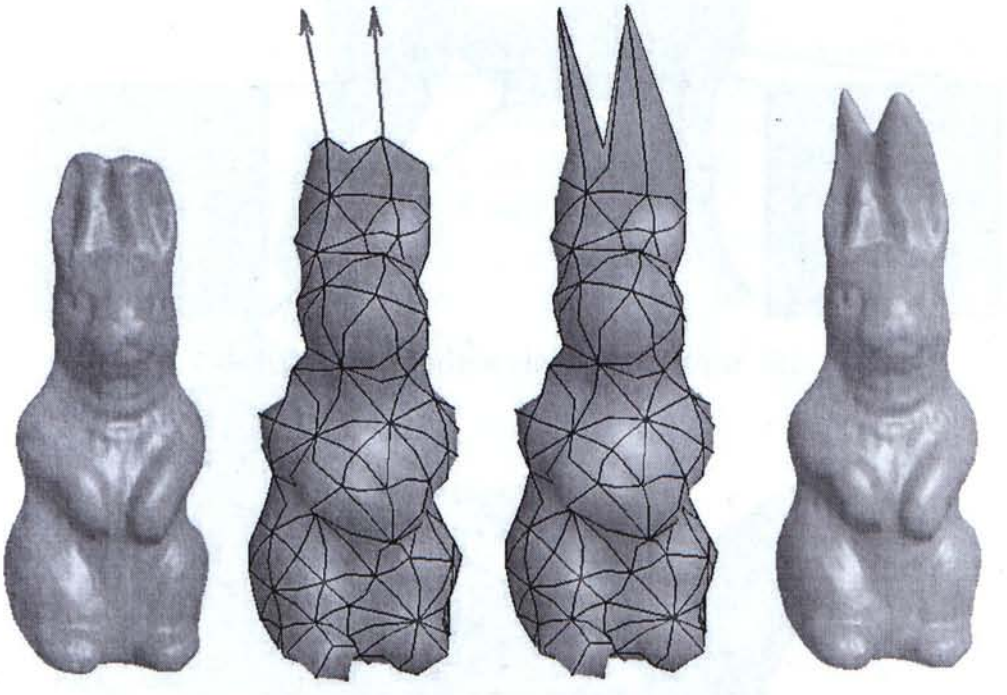


(b) wavelet analysis with constraints

Figure 29. Elimination of gaps at the joints of different resolution parts of a model

**Experiment 10. Multi-resolution geometric editing of Bunny with Loop subdivision wavelet transform:**

In Figure 30, (a) is the original model Bunny; (b) is the base mesh of bunny; (c) is the edited base mesh, where the two ears are stretched; (d) is the edited result of the model.



(a) (b) (c) (d)

Figure 30. Multi-resolution geometric editing of model Bunny.

# 5 Conclusions and future work

In the first part of our thesis, we have proved the explicit formulae of both the general recursion scheme and the de Boor algorithms for computing non-uniform B-spline curves and their derivatives of arbitrary degree by mathematical induction; and then an explicit matrix representation for B-spline curves of arbitrary degree is also given. The explicit matrix formula transforms non-uniform B-spline curves of arbitrary degree to the polynomial space spanned by the common power basis and enables efficiently computing the B-spline curves and surfaces and their derivatives by using Horner' s schema. The explicit computing formulae developed in this paper are very useful for the evaluation and the conversion of curves and surfaces in CAD/CAM systems.

Based on lifting scheme, we construct a biorthogonal wavelet of Doo-Sabin subdivision and put forward geometrically constrained wavelet constructions with different subdivision including Loop, Catmull-Clark and Doo-Sabin. In Doo-Sabin subdivision, all the old control points disappear after the subdivision process. Our work solve this problem, the corresponding lazy wavelet transform can be constructed and translated into a sequence of simple local operations easily, without the extended subdivision rules. It means that this method is more flexible to different kind of models.

The construction of the lifting scheme designed for geometric constraints extends the modeling ability of the lifting-based subdivision wavelet constructions. With our geometrically constrained wavelet transforms, the 3D models analyzed can preserve user specified discrete and continuous constraints like positions, normals, $d^{th}$ derivatives and isoparametric curves. Furthermore, by means of elaborately-designed position and normal constraints, our approach can also deal with sharp features in multiresolution modeling. Compared to the previous multi-resolution modeling methods, the differences are the the previous methods need to change the subdivision rules to keep the sharp features. This made the subdivision-based wavelet transform algorithms more complicated and difficult to implement. Our method use the lifting scheme instead of changing any subdivision rules, and our method can deal with high order derivative constraints, as the experiments shown in pervious chapter.

Our biorthogonal subdivision wavelet construction method, which is based on geometrically constrained lifting operations without any change of the subdivision rules, can also be adopted for other subdivision-wavelet-based multi-resolution modeling algorithms. This will be our future work.

# Appendix A

## Matlab programs for the matrix [Ak] in Equation (15)

```
% matrixA.m
function f=matrixA(k,i)
R = sym(zeros(k+1,k+1));
for j=0:k
    for m=0:k
    R(j+1,m+1) = AmatrixEx(k,i,j,m);
    end
end
f=simplify(R);

% AmatrixEx.m
function f = AmatrixEx(k,i,j,m)
if j ==0
    if m ==0
        f = 1;
        for a =1:k
            f = f* beta(i-k,a,0,k,i);
        end
    else
        f = sumF(m,k,j,i);
    end
else
if m==0
  f = ffunction(0,i-k,1,k,j,m,k,i)/fibonacci(j);
    else
        f = sumF(m,k,j,i)/fibonacci(j);
end
end
factor(f);
simplify(f);

% sumF.m
function f = sumF(m, k,j, i)
cycle = m;
length = k-m+1;
l =0;
if j==0
    Element = sym(zeros(1,length));
    R =   sym(zeros(1,length));
```

```
    for am=1:length
  Element(am) = ffunction(0,i-k+m,1,am-1, 0,m,k,i);
    end
    for a0=1:cycle
        for a1 = a0:length+a0-1
            R(a1-l)=0;
            for am=a0:a1
                R(a1-l)=  R(a1-l)+Element(am-l)*alpha(i-k+m-a0,  am,  0,k,i)*ffunction(0,
i-k+m-a0,am+1,a1-1,0,m,k,i);
            end
        end
       l=l+1;
        Element=R;
    end
        f=R(length);

else
    Element = sym(zeros(length,j+1));
    R =  sym(zeros(length,j+1));
    for am=1:length
        for bm =0:j
         Element(am,bm+1) = ffunction(0,i-k+m,1,am-1, bm,m,k,i);
        end
    end
    for a0=1:cycle
        for a1 = a0:length+a0-1
            for bm =0:j
            R(a1-l,bm+1)=0;
            end
            for am=a0:a1
                for bm=0:j
                    for bm1 = 0:bm
                        for bm2 = 0:bm-bm1
                            bm3 = bm-bm1-bm2;

R(a1-l,bm+1)=R(a1-l,bm+1)+fibonacci(bm)*Element(am-l,bm1+1)*alpha(i-k+m-a0,          am,
bm2,k,i)*ffunction(0,
i-k+m-a0,am+1,a1,bm3,m,k,i)/(fibonacci(bm1)*fibonacci(bm2)*fibonacci(bm3));
                        end
                    end
                end
            end
        end
        l=l+1;
        Element=R;
```

```
    end
    f=R(length,j+1);
end

% ffunction.m
function f = ffunction(x,y,b,c,xi,m,k,li)
tf=0;
y = y;
if b>c
    if xi==0
        f = 1;
    end
    if xi >=1
        f = 0;
    end
else
    if xi ==0
        tf = 1;
        for b0=b:c
            tf = tf*beta(y,b0,0,k,li);
        end
    else
        a = c-b;
        XI=[1:a+1];
        for i=1:a
            XI(i) = 0;
        end
        while XI(1)<=xi
            XI(a+1) = xi;
            for i=1:a
                XI(a+1) = XI(a+1)-XI(i);
            end
            p=fibonacci(xi);
            for i=1:a+1
                temp = b+i-1;
                if m==0 p = p*beta(y,temp, XI(i),k,li)/fibonacci(XI(i));
                else p = p*beta(y,temp, XI(i),k,li)/fibonacci(XI(i));
                end
            end
            tf=tf+p;
            tempS = 0;
             for i=1:a+1
                tempS = tempS + XI(i);
                if tempS>=xi
                    if i ==1 XI(i) = xi+1; break;
```

```
                    else
                        XI(i-1)=XI(i-1)+1;
                        for j = i:a+1
                            XI(j) = 0;
                        end
                        break;
                    end
                end
            end
        end
    end
  f =tf;
% alpha.m
function f = alpha(j, r, xi,k,i)
t = sym('ti');
t2 = sym('ti1');
if xi==0
    f = (T(i)-T(j+r))/(T(j+k+1)-T(j+r));
else
    if xi ==1
        f = (T(i+1)-T(i))/(T(j+k+1)-T(j+r));
    else
        f = 0;
    end
end

% beta.m
function f = beta(j, r, xi,k,i)
t = sym('ti');
t2 = sym('ti1');
if xi==0
    f = (T(j+k+1)-T(i))/(T(j+k+1)-T(j+r));
else
    if xi ==1
        f = -(T(i+1)-T(i))/(T(j+k+1)-T(j+r));
    else
        f = 0;
    end
end

% fibonacci.m
function f = fibonacci(x)
f = 1;
if x<1
    f = 1;
```

```
else
    while x>1
        f = f*x;
        x = x-1;
    end
end

% T.m
function f = T(i)
T=sym('[t0 t1 t2 t3 t4 t5 t6 t7 t8 t9 t10 t11 t12 t13 t14]');
f=T(i+1);
```

# Appendix B

$$[A^4]=\begin{bmatrix} A^4_{0,0} & A^4_{0,1} & A^4_{0,2} & A^4_{0,3} & 0 \\ -4A^4_{0,0} & A^4_{1,1} & A^4_{1,2} & A^4_{1,3} & 0 \\ 6A^4_{0,0} & A^4_{2,1} & A^4_{2,2} & A^4_{2,3} & 0 \\ -4A^4_{0,0} & A^4_{3,1} & A^4_{3,2} & A^4_{3,3} & 0 \\ A^4_{0,0} & A^4_{4,1} & A^4_{4,2} & A^4_{4,3} & A^4_{4,4} \end{bmatrix}$$

$$A^4_{0,0}=\frac{(t_{i+1}-t_i)^3}{(t_{i+1}-t_{i-3})(t_{i+1}-t_{i-2})(t_{i+1}-t_{i-1})}$$

$$A^4_{0,1}=\frac{1}{t_{i+1}-t_{i-1}}\left[\frac{(t_i-t_{i-3})(t_{i+1}-t_i)^2}{(t_{i+1}-t_{i-2})(t_{i+1}-t_{i-3})}+\frac{(t_i-t_{i-1})(t_{i+2}-t_i)^2}{(t_{i+2}-t_{i-2})(t_{i+2}-t_{i-1})}+\frac{(t_i-t_{i-2})(t_{i+1}-t_i)(t_{i+2}-t_i)}{(t_{i+1}-t_{i-2})(t_{i+2}-t_{i-2})}\right]$$

$$A^4_{0,2}=\frac{1}{t_{i+1}-t_{i-1}}\left[\frac{(t_i-t_{i-2})(t_i-t_{i-1})}{t_{i+2}-t_{i-2}}+\frac{(t_i-t_{i-1})^2(t_{i+3}-t_i)}{(t_{i+3}-t_{i-1})(t_{i+2}-t_{i-1})}+\frac{(t_i-t_{i-2})^2(t_{i+1}-t_i)}{(t_{i+1}-t_{i-2})(t_{i+2}-t_{i-2})}\right]$$

$$A^4_{0,3}=\frac{(t_i-t_{i-1})^3}{(t_{i+1}-t_{i-1})(t_{i+2}-t_{i-1})(t_{i+3}-t_{i-1})}$$

$$\begin{aligned} A^4_{1,1}=&\frac{4(t_i-t_{i+1})}{(t_{i-1}-t_{i+1})(t_{i-2}-t_{i+1})(t_{i-2}-t_{i+2})(t_{i-1}-t_{i+2})(t_{i-3}-t_{i+1})}\{(t_{i-3}- \\ &t_{i+1})\left[t_i^2(t_{i+2}+t_{i+1}+t_{i-1}-t_{i-2})-t_{i+2}(t_it_{i+1}-t_i^2-t_{i-2}t_{i-1})\right] \\ &+t_{i+2}(t_i-t_{i+2})(t_{i-2}t_i+t_5^2-t_{i-1}t_i)+(t_{i-3}-t_i)(t_{i-2}t_{i+1}t_{i+2}-t_{i-2}t_{i-1}t_i \\ &-t_{i-1}t_{i+1}t_{i+2}-t_it_{i+1}t_{i+2})+(t_{i+1}-t_i)(t_{i-3}t_{i-2}t_{i-1}+t_it_{i+2}^2)\} \end{aligned}$$

$$A^4_{1,3}=\frac{4(t_i-t_{i-1})^2(t_{i+1}-t_i)}{(t_{i+1}-t_{i-1})(t_{i+2}-t_{i-1})(t_{i+3}-t_{i-1})}$$

$$A^4_{2,3}=\frac{6(t_i-t_{i-1})(t_{i+1}-t_i)^2}{(t_{i+1}-t_{i-1})(t_{i+2}-t_{i-1})(t_{i+3}-t_{i-1})}$$

$$A^4_{3,3}=\frac{4(t_{i+1}-t_i)^3}{(t_{i+1}-t_{i-1})(t_{i+2}-t_{i-1})(t_{i+3}-t_{i-1})}$$

$$A^4_{4,3}=-\frac{(t_{i+1}-t_i)^3}{(t_{i+3}-t_{i-1})(t_{i+3}-t_i)}\left[\frac{1}{(t_{i+2}-t_i)}+\frac{1}{(t_{i+2}-t_{i-1})}\right]-A^4_{4,4}-\frac{1}{4}A^4_{3,3}$$

$$A_{2,1}^4 = \frac{-6(t_{i+1}-t_i)^2}{(t_{i-2}-t_{i+1})(t_{i-1}-t_{i+1})(t_{i-2}-t_{i+2})}\left[t_i + \frac{t_i(t_{i-2}+t_{i+2})}{(t_{i-3}-t_{i+1})}\right.$$
$$\left. + \frac{t_{i+1}(t_{i+2}-t_i)(t_{i-3}-t_{i+1}-t_{i-2}) + t_{i-3}t_{i-2}(t_{i-1}-t_i)}{(t_{i-3}-t_{i+1})(t_{i-1}-t_{i+2})}\right]$$

$$A_{3,1}^4 = 4A_{0,0}^4 + \frac{4(t_{i+1}-t_i)^3}{(t_{i+2}-t_{i-2})(t_{i+1}-t_{i-1})}\left[\frac{1}{(t_{i+2}-t_{i-1})} + \frac{1}{(t_{i+1}-t_{i-3})}\right]$$

$$A_{4,1}^4 = -A_{0,0}^4 - \frac{(t_{i+1}-t_i)^3}{(t_{i+2}-t_{i-2})}\left[\frac{1}{(t_{i+1}-t_{i-2})(t_{i+1}-t_{i-1})} + \frac{1}{(t_{i+2}-t_{i-1})(t_{i+1}-t_{i-1})} + \frac{1}{(t_{i+2}-t_i)(t_{i+2}-t_{i-1})}\right]$$

$$A_{1,2}^4 = \frac{-4(t_i-t_{i+1})}{(t_{i-2}-t_{i+2})(t_{i-1}-t_{i+1})(t_{i-1}-t_{i+2})(t_{i-1}-t_{i+3})}\{t_{i-1}t_{i+2}t_{i+3}$$
$$+ t_i^2(t_{i-2}+t_{i+2}-t_{i+3}) - 2t_{i-1}t_i(t_{i-2}-t_i) + [t_it_{i+2}t_{i+3}(t_i-t_{i+3})$$
$$+ (t_{i-2}-t_i)(t_{i-1}t_it_{i+3} + t_{i+1}t_{i+2}t_{i+3} + t_{i-1}t_it_{i+2}) + t_{i-1}^2(t_{i-2}-t_i)^2$$
$$- t_{i-2}t_{i-1}(t_{i+1}-t_i)(t_{i+2}+t_{i+3})]/(t_{i-2}-t_{i+1})\}$$

$$A_{2,2}^4 = \frac{-6(t_i-t_{i+1})^2}{(t_{i-2}-t_{i+2})(t_{i-1}-t_{i+1})(t_{i-1}-t_{i+2})}\{t_i + [t_{i-1}(t_{i-2}-t_i)(t_{i-1}-t_{i+3})$$
$$+ (t_{i-2}-t_{i+1})(t_{i-2}t_{i-1} - t_{i-2}t_i - t_it_{i+2}) + t_{i-1}t_{i+2}(t_{i-2}-t_i)$$
$$+ t_{i+2}t_{i+3}(t_i-t_{i+1})]/[(t_{i-2}-t_{i+1})(t_{i-1}-t_{i+3})]\}$$

$$A_{3,2}^4 = -A_{3,3}^4 - \frac{4(t_{i+1}-t_i)^3}{(t_{i+1}-t_{i-2})(t_{i+2}-t_{i-1})}\left[\frac{1}{(t_{i+1}-t_{i-1})} + \frac{1}{(t_{i+2}-t_{i-2})}\right]$$

$$A_{4,2}^4 = \frac{1}{4}A_{3,3}^4 - A_{4,1}^4 - A_{0,0}^4 + \frac{(t_{i+1}-t_i)^3}{(t_{i+2}-t_i)(t_{i+3}-t_{i-1})}\left[\frac{1}{t_{i+2}-t_{i-1}} + \frac{1}{t_{i+3}-t_i}\right]$$

$$A_{4,4}^4 = \frac{(t_{i+1}-t_i)^3}{(t_{i+2}-t_i)(t_{i+3}-t_i)(t_{i+4}-t_i)}$$

# Appendix C

$$\beta(n_i)=\frac{8}{5}\alpha(n_i)-\frac{3}{5} \quad \gamma_i \equiv \frac{1-\alpha(n_i)}{n_i} \quad \delta_i \equiv \frac{1-\beta(n_i)}{n_i}$$

$$P=\begin{bmatrix} P_1 \\ P_2 \end{bmatrix}$$

$$P_1=\left[\begin{array}{c|cccc|c|ccccc}
\alpha_0 & \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 & \cdots & & \gamma_{n-2} & \gamma_{n-1} & \gamma_n & 0 \\
\frac{3}{8} & \frac{3}{8} & \frac{1}{8} & 0 & 0 & \cdots & 0 & 0 & 0 & \frac{1}{8} & \\
\hline
\frac{3}{8} & \frac{1}{8} & \frac{3}{8} & \frac{1}{8} & 0 & \cdots & & & & & \\
\frac{3}{8} & 0 & \frac{1}{8} & \frac{3}{8} & \frac{1}{8} & \cdots & & 0 & & & 0 \\
\frac{3}{8} & 0 & 0 & \frac{1}{8} & \frac{3}{8} & \cdots & & & & & \\
\vdots & 0 & 0 & 0 & \frac{1}{8} & \ddots & & & & & \\
\hline
\frac{3}{8} & & & & & \cdots & \frac{1}{8} & 0 & 0 & 0 & \\
\frac{3}{8} & & 0 & & & \cdots & \frac{3}{8} & \frac{1}{8} & 0 & 0 & 0 \\
\frac{3}{8} & & & & & \cdots & \frac{1}{8} & \frac{3}{8} & \frac{1}{8} & 0 & \\
\frac{3}{8} & & & & & \cdots & 0 & \frac{1}{8} & \frac{3}{8} & \frac{1}{8} &
\end{array}\right]$$

$$P_2 = \begin{bmatrix}
\frac{1}{8} & \frac{3}{8} & 0 & & & 0 & \frac{3}{8} & \frac{1}{8} & 0 & 0 & 0 & 0 \\
\gamma_0 & \alpha_1 & \gamma_2 & & & 0 & \gamma_n & \gamma_{n+1} & \gamma_{n+2} & \gamma_{n+3} & 0 & 0 \\
\frac{1}{8} & \frac{3}{8} & \frac{3}{8} & & & 0 & 0 & 0 & 0 & \frac{1}{8} & 0 & 0 \\
\gamma_0 & \gamma_1 & 0 & & & \gamma_{n-1} & \alpha_n & \gamma_{n+1} & 0 & 0 & \gamma_{n+4} & \gamma_{n+5} \\
\frac{1}{8} & 0 & 0 & \ddots & & \frac{3}{8} & \frac{3}{8} & 0 & 0 & 0 & 0 & \frac{1}{8} \\
0 & \frac{3}{8} & 0 & & 0 & 0 & \frac{1}{8} & \frac{3}{8} & \frac{1}{8} & 0 & 0 & 0 \\
0 & \frac{3}{8} & 0 & & \ddots & 0 & 0 & \frac{1}{8} & \frac{3}{8} & \frac{1}{8} & 0 & 0 \\
0 & \frac{3}{8} & 0 & & & 0 & 0 & 0 & \frac{1}{8} & \frac{3}{8} & \frac{1}{8} & 0 \\
0 & \frac{1}{8} & 0 & & & 0 & \frac{3}{8} & \frac{3}{8} & 0 & 0 & \frac{1}{8} & 0 \\
0 & 0 & 0 & & & 0 & \frac{3}{8} & \frac{1}{8} & 0 & 0 & \frac{3}{8} & \frac{1}{8} \\
0 & 0 & 0 & & & \frac{1}{8} & \frac{3}{8} & 0 & 0 & 0 & \frac{1}{8} & \frac{3}{8}
\end{bmatrix}$$

$$Q = \begin{bmatrix}
\delta_0 & \overbrace{\cdots} & \delta_0 & \delta_0 & \overbrace{\cdots} & \delta_0 & & 0 & & & 0 & \\
1 & & & & & 0 & & & & & & \\
& \ddots & & & & & & & & & & \\
& & 1 & & & & & 0 & & & 0 & \\
& & & 1 & & & & & & & & \\
& & & & \ddots & & & & & & & \\
0 & & & & & 1 & & & & & & \\
& & & 0 & & & & 0 & & & 0 & 0 & 1 \\
\delta_{n+2} & \cdots & 0 & 0 & \cdots & 0 & \delta_{n+2} & \delta_{n+2} & \delta_{n+2} & 0 & 0 & 0 & \delta_{n+2} & 0 & \delta_{n+2} \\
& & & 0 & & & & & & 0 & & & 1 & 0 & 0 \\
0 & \cdots & 0 & 0 & \cdots & \delta_{n+4} & 0 & 0 & 0 & \delta_{n+4} & \delta_{n+4} & \delta_{n+4} & 0 & \delta_{n+4} & \delta_{n+4} \\
& & & 0 & & & & & & 0 & & & 0 & 1 & 0 \\
& & & & & & 1 & & & & & 0 & & & \\
& & & & & & & 1 & & & & & & & \\
& & 0 & & & & & & 1 & & & & & 0 & \\
& & & & & & & & & 1 & & & & & \\
& & & & & & & & & & 1 & & & & \\
& & & & & & 0 & & & & & 1 & & &
\end{bmatrix}$$

# Appendix D

The normal constraints function for Lagrange multiplier is:

$$F=\sum_{\ell=0}^{n+8}\left[\sum_{j=0}^{n+5}(g_{j\ell})^2+\lambda_\ell\left(\sum_{r=0}^{n+5}B_r s_{r\ell}-c_\ell\right)+\lambda_\ell^v\left(\sum_{r=0}^{n+5}\frac{\partial B_r}{\partial v}s_{r\ell}-c_\ell^v\right)+\lambda_\ell^w\left(\sum_{r=0}^{n+5}\frac{\partial B_r}{\partial w}s_{r\ell}-c_\ell^w\right)\right]$$

Let $F=\sum_{\ell=0}^{n+8}F_\ell$, and

$$F_\ell=\sum_{j=0}^{n+5}(g_{j\ell})^2+\lambda_\ell\left(\sum_{r=0}^{n+5}B_r s_{r\ell}-c_\ell\right)+\lambda_\ell^v\left(\sum_{r=0}^{n+5}\frac{\partial B_r}{\partial v}s_{r\ell}-c_\ell^v\right)+\lambda_\ell^w\left(\sum_{r=0}^{n+5}\frac{\partial B_r}{\partial w}s_{r\ell}-c_\ell^w\right)$$

$$=\sum_{j=0}^{n+5}\left[(g_{j\ell})^2+\lambda_\ell B_j s_{j\ell}+\lambda_\ell^v\frac{\partial B_j}{\partial v}s_{j\ell}+\lambda_\ell^w\frac{\partial B_j}{\partial w}s_{j\ell}\right]-\lambda_\ell c_\ell-\lambda_\ell^v c_\ell^v-\lambda_\ell^w c_\ell^w$$

$$=\sum_{j=0}^{n+5}\left[\left(<\phi_j,\psi_\ell>+\sum_{k=0}^{n+5}<\phi_k,\phi_j>s_{k,\ell}\right)^2+\lambda_\ell B_j s_{j\ell}+\lambda_\ell^v\frac{\partial B_j}{\partial v}s_{j\ell}+\lambda_\ell^w\frac{\partial B_j}{\partial w}s_{j\ell}\right]-\lambda_\ell c_\ell-\lambda_\ell^v c_\ell^v-\lambda_\ell^w c_\ell^w$$

$$\frac{\partial F}{\partial s_{i\ell}}=\frac{\partial F_\ell}{\partial s_{i\ell}}$$

$$=2\sum_{j=0}^{n+5}<\phi_i,\phi_j>\left(<\phi_j,\psi_\ell>+\sum_{k=0}^{n+5}s_{k,\ell}<\phi_k,\phi_j>\right)+\lambda_\ell B_i+\lambda_\ell^v\frac{\partial B_i}{\partial v}+\lambda_\ell^w\frac{\partial B_i}{\partial w}=0$$

$$2\sum_{j=0}^{n+5}<\phi_i,\phi_j>\left(\sum_{k=0}^{n+5}s_{k,\ell}<\phi_k,\phi_j>\right)+\lambda_\ell B_i+\lambda_\ell^v\frac{\partial B_i}{\partial v}+\lambda_\ell^w\frac{\partial B_i}{\partial w}=-2\sum_{j=0}^{n+5}<\phi_i,\phi_j><\phi_j,\psi_\ell>$$

where $\ell=0,1,\cdots,n+8;\ j=0,1,\cdots,n+5.$

## References


[1] Bertram, M. 2004. Biorthogonal Loop-subdivision wavelets. Computing 72, 1-2, 29 - 39.

[2] Bertram, M., Duchaineau, M.A., and Hamann, B. 2000. Bicubic subdivision-surface wavelets for large-scale isosurface representation and visualization. In Proceedings of Visualization 2000. 389 - 396.

[3] Böhm, W., 1980. Inserting new knots with B-spline curves. Computer-Aided Design 12 (4), 199 - 201.

[4] Boor, C. D., Höllig, K., AND RIEMENSCHNEIDER, S. Box splines. Springer Verlag, 1994.

[5] Boor, C. D., On the evaluation of box splines. Numer. Algorithms, 5:5-23, 1993.

[6] Catmull E., Clark J., Recursively Generated B-spline Surfaces on Arbitrary Topological Meshes. Comput. Aided Des. 10, 350-355, 1978

[7] Chang, G., 1982. Matrix foundation of Bézier technique. Computer-Aided Design 14 (6), 350 - 354.

[8] Chen, Z., Luo, X., Tan, L., Ye, B., Chen, J., 2008. Progressive Interpolation based on Catmull-Clark Subdivision Surfaces. Computer Graphics Forum 27 (7), 1823-1827.

[9] Choi, B.K., Yoo, W.S., Lee, C.S., 1990. Matrix representation for NURBS curves and surfaces. Computer-Aided Design 22 (4), 235 - 240.

[10] Cohen, E., Riesenfeld, R.F., 1982. General matrix representation for Bézier and B-spline curves. Computers in Industry 3, 9 - 15.

[11] Cox, M. G., 1972. The numerical Evaluation of B-splines. J. Inst. Maths. Applics., 10:134-149.

[12] De Boor, C.W., 1972. On calculating with B-splines. J. Approx. Theory 6, 50 - 62.

[13] De Boor, C., and Devore, R. Approximation by smooth multivariate splines. Trans Amer Math Soc 276, 775 - 788, 1983

[14] Ding, Q.L., Davies, B.J., 1987. Surface Engineering Geometry for Computer Aided Design and Manufacture. Ellis Horwood.

[15] Doo D., Sabin M., Behaviors of Recursive Division Surfaces near Extraordinary Point. Computer-Aided Design 10, 6, 356-360, 1978

[16] Farin, Gerald E. 2002. Curves and Surfaces for Computer Aided

Geometric Design: A Practical Guide. Morgan Kaufmann Publishers, 5th edition

[17] Frederickson, P.O., Triangular spline interpolation. Rpt.6 70, Lakehead Univ, 1970.

[18] Grabowshi, H., Li, X., 1992. Coefficient formula and matrix of nonuniform B-spline functions. Computer-Aided Design 24 (12), 637 - 642.

[19] Hoschek, J. and Lasser, D., 1993. Fundamentals of Computer Aided Geometric Design. Translated by Schumaker, Larry L., A K Peters, Wellesley.

[20] Lai M.J., Fortran Subroutines For B-Nets of Box Splines on Three-and Four-Directional Meshes. Numerical Algorithms, 2:33-38, 1992

[21] Li Denggao, Qin Kaihuai, Sun Hanqiu, Unlifted Loop Subdivision Wavelets, Proceedings of Pacific Graphics '04, pp. 25-33, Korea, 2004

[22] Li Denggao, Qin K., Sun H., Curve Modeling with Constrained B-Spline Wavelets. Computer Aidd Geometric Design 22(1), 45-56. 2005

[23] Lin, S., Luo, X., You, F., Li, Z., 2008. Deducing Interpolating Subdivision Schemes from Approximating Subdivision Schemes. ACM Transactions on Graphics 27 (5), 146:1-146:7.

[24] Liu, L., Wang, G., 2002. Explicit matrix representation for NURBS curves and surfaces. Computer Aided Geometric Design, 19(6), 409-419.

[25] Loop, C.T., Smooth Subdivision Surfaces Based on Triangles. Master' s thesis. University of Utah, 1987.

[26] Lounsbery J.M., Multiresolution Analysis for Surfaces of Arbitrary Topological Type, Ph.D. Thesis, Department of Mathematics, University of Washington, 1994.

[27] Pan, R. J., 2001. Explicit Matrix Representation for NURBS Curves and Surfaces and Its Algorithm. Chinese Journal of Computers, 24 (04), 358-366.

[28] Qin, K., General Matrix Representations for B-splines. The Visual Computer 16(33), 177-186, 2000.

[29] Sabin, M.A., The Use of Piecewise Forms for the Numerical Representation of Shape. Dissertation, MTA Budapest, 1977.

[30] Schröder, P., Sweldens, W., 1995. Spherical Wavelets: Efficiently Representing Functions on the Sphere. In: ACM SIGGRAPH 1995 Conference Proceedings. Los Angeles, California, USA, pp. 161-172.

[31] Schumaker, Larry L. , Spline Functions: Basic Theory (3rd Ed.). Cambridge University Press, 2007.

[32] Shi, Z., Lin, S., Luo, X., Wang, R., 2008. Interpolatory and Mixed Loop Schemes. Computer Graphics Forum 27 (7), 1829-1835.

[33] Stam J., Evaluation of Loop Subdivision Surfaces. Proceedings of ACM SIGGRAPH 1998

[34] Stam J., Exact Evaluation of Catmull-Clark Subdivision Surfaces at Arbitray Parameter Values. SIGGRAPH 1998

[35] Sweldens, W., 1996. The Lifting Scheme: A Custom-Design Construction of Biorthogonal Wavelets. Applied and Computational Harmonic Analysis 3 (2), 186-200.

[36] Sweldens, W., 1997. The Lifting Scheme: A Construction of Second Generation Wavelets. SIAM Journal on Applied Mathematics 29 (2), 511-546.

[37] Wang H., Qin K., Tang K., Efficient Wavelet Construction with Catmull-Clark Subdivision, Visual Comput 22: 874-884, 2006

[38] Wang H., Qin K., Precise Evaluation of Uniform Doo-Sabin Surfaces. Progress in Natural Science 13, 5, 391-396, 2003.

[39] Wang, H., Qin, K., and Sun, H. $\sqrt{3}$ -Subdivision-based biorthogonal wavelets. IEEE Transactions on Visualization and Computer Graphics 13, 5, 914 - 924. 2007.

[40] Wang H., Tang K., Qin K., Biorthogonal wavelets based on gradual subdivision of quadrilateral meshes. Computer Aided Geometric Design, vol. 25, no. 9, pp. 816 - 836, 2008. 2008.

[41] Zhang, H., Qin, G., Qin, K., and Sun, H. 2008. A biorthogonal wavelet approach based on dual subdivision. Computer Graphics Forum 27, 7, 1815 - 1822.